QL Today

Volume 12
Issue 3
March-May
2008

ISSN 1432-5454

The Magazine about QL, QDOS, Sinclair Computers, SMSQ...

QL2K

Dilwyn Jones reports about the latest Version

More I/O

for QL and Emulators

Ian Burkinshaw comes up with a flexible solution

Assembling SMSQ/E with GWASS

George Gwilt explains in Detail "how to"

More interesting GPS-related stuff from Hugh Rooms:

The Mercator Map Projection

Norman Dunbar tackles the

Programming of the Pointer Environment

in Part 20 of his Assembler Series

... and much more - all inside this Issue!

www.QLToday.com

Contents

Advertisers

in alphabetical order

QL Today

ISSN 1432-5454

German office & Publisher:

Jochen Merz Software
Kaiser-Wilhelm-Str. 302
47169 Duisburg
Germany

Tel. +49 203 502011
Fax +49 203 502012
email: JMerz@j-m-s.com
email: QLToday@j-m-s.com

English office:

Q Branch
20 Locks Hill
Portslade
BN41 2LB
United Kingdom

Tel. +44 1273 430501
Mobile +44 7836 745501
Fax +44 1273 430501
email: qbranch@qbranch.demon.co.uk
email: QLToday@j-m-s.com

Editor:

Geoff Wicks
5b Wordsworth Avenue
Sinfin
Derby DE24 9HQ
United Kingdom

Tel. +44 1332 271366
email: gwicks@beeb.net
email: QLToday@j-m-s.com

Co-Editor:

Bruce Nicholls
38 Derham Gardens
Upminster
Essex RM14 3HA
United Kingdom

Tel +44 20 71930539
Fax +44 870 0568755
email: qltoday@q-v-d.demon.co.uk
email: QLToday@j-m-s.com

QL *Today* is published five times a year, our volume begins on beginning of June. Please contact the German or English office for current subscription rates or visit our homepage **www.QLTODAY.com**.

We welcome your comments, suggestions and articles. YOU make **QL *Today*** possible. We are constantly changing and adjusting to meet your needs and requirements. Articles for publication should be on a 3.5" disk (DD or HD) or sent via Email. We prefer ASCII, Quill or text87 format. Pictures may be in _SCR format, we can also handle GIF or TIF or JPG. To enhance your article you may wish to include Saved Screen dumps. PLEASE send a hardcopy of all screens to be included. Don't forget to specify where in the text you would like the screen placed.

If you need more information about the UNZIP program which is used by our BOOT program to unpack the files, we suggest that you visit Jonathan Hudsons web site where you find more information about lots of interesting QDOS software and INFOZIP at **www.bigfoot.com/~jrhudson/**

The deadline for the next issue is the 18th of May 2008

Editorial

by Geoff Wicks

The first issue of a new year is a time to take stock. 2008 could be a difficult QL year, especially in the UK.

Our first news report of the year concerns reduced UK trader activity. In recent years QL traders have faced increasing difficulties and it is becoming less and less worthwhile for them to attend shows. It is not just a question of finance, but also of time. We could well see a further reduction during 2008.

In this climate QL Today cannot expect to remain unscathed. We are now looking at possible changes to ensure the short and medium term future of the magazine.

I heard a story over the grapevine that a respected trader is about to close his business for personal reasons, with a final, month long, closing down sale. Actually this is a true story, but it happened many years ago. The trader concerned was a guy called Dilwyn Jones - you may have heard of him.

Thirteen years on Dilwyn is no forgotten figure from the past, but has been a distinguished first editor of this magazine, runs the most popular QL website and has a prolific, and much valued, software output.

A year before closing down Dilwyn had had the dubious honour of switching off the lights in QL World. His advert was the last page of the last issue of the magazine. The closure of QL World was seen by many as a harbinger of the imminent death of the QL. Two replacements, QReview and IQLR, were short-lived, but eventually out of the ashes QL Today emerged. We have survived longer than any other QL publication with the exception of the Quanta Magazine.

When I closed the commercial side of Just Words! last year, there was no loss to the QL community other than some QL Today advertising revenue. Instead there was a rewritten and expanded website that has been further expanded this year.

In difficult times we have to stand firm and not be frightened of change. The days of QL commercial software are probably at an end, but programming continues. The QL market has become too small for new hardware, but at least one trader runs a profitable business in second hand material. Show attendance may be in decline, but we can keep in touch via the internet.

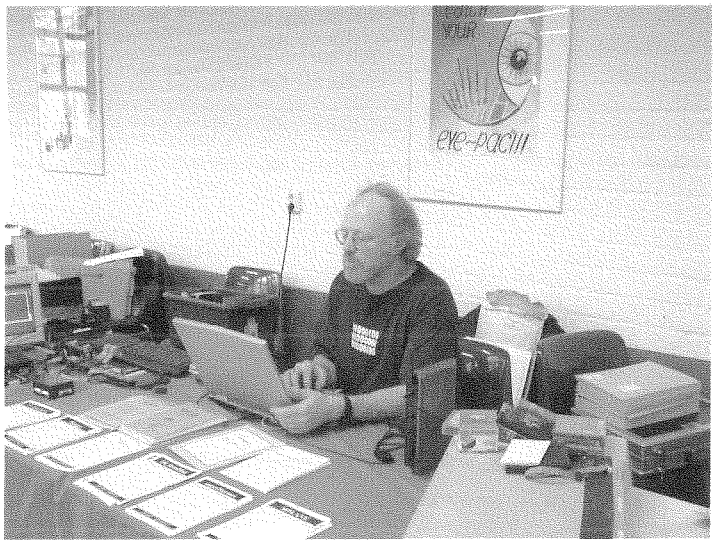
One bright spot is that QL Today understands - strictly off the record - that by the time you are reading this Quanta will have some good news for its members, although these are still falling in number.

Quanta now has under 200 members, but its structure, activities and thinking remain largely rooted in the time when it had over 2,000. Can Quanta adapt to changing circumstances? There are about 40 reasonably active people within Quanta and given a radical reshaping around the needs of these members it could remain a powerful force within the QL community.

For some years Quanta has been the victim of its members' benign neglect, but the immediate danger of its closure in 2009 has been averted. Next year Quanta plans a major celebration of the QL's centenary. Let us set our sights on this celebration and use this to get through the dark days of 2008.

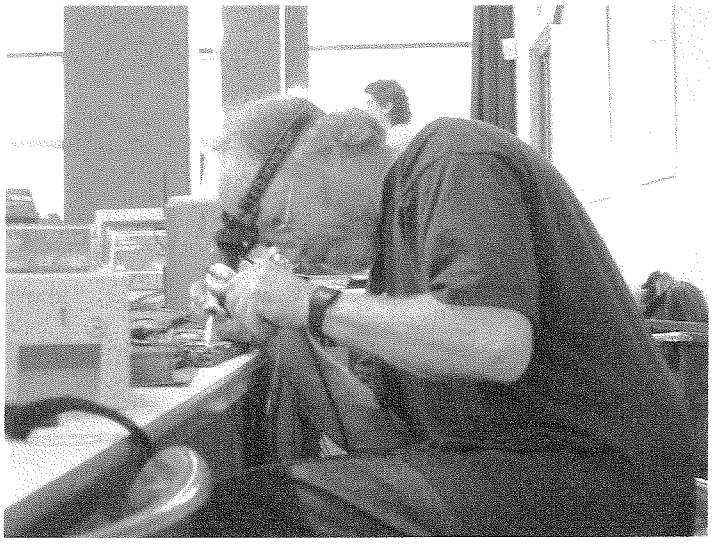
TFS bows out

QL trader Tony Firshman has asked QL Today to withdraw his advertising from the magazine, which he had booked for the whole of the present volume. In an email to QL Today he explained that increasing demands on his time means he is no longer able to guarantee the quality of service to QL users an active advertising presence would demand. One order from a client took him 5 weeks to fulfil.



Tony Firshman does not intend to close TF Services or to leave the QL community, but from now on he will maintain a lower profile.

There has been some falling off in QL trader activity in recent years and QL Today believes this could continue during 2008. This changing trading pattern could affect the magazine and the QL Today team are currently looking at its administration, distribution and financing to ensure the short and medium term future. (A full report can be found at the end in this issue.)



At the end of last year Jochen Merz software opened bank accounts in several countries, including the UK, to facilitate transactions with QL-ers living outside Germany.

Tony Firshman is the longest serving of the current QL traders and the origins of TF Services go back to the earliest days of the QL. The company has become known for several innovative QL hardware products. One of its earliest lines, launched in 1985, was a power cleaner to reduce QL crashes by filtering out mains power spikes and surges. A bar code reader followed in 1988, but it was not until the 1990's that TF Services became known for a number of products that revolutionised the QL by improving its stability and reliability.

In 1989 QView launched the Minerva ROM and TF Services was the hardware specialist backing this venture. TF Services took over the sales in 1991 and in the same year launched a MKII version. The following year they launched the Hermes chip to replace the 8049 co-processor which gave the QL more reliable serial communications. This was followed in 1995 by a SuperHermes board giving a fast serial port with mouse and keyboard interfaces. A final version of SuperHermes, SuperHermes Lite, was released in 1997.

The same year TF Services announced the RomDisq, a flash memory card long before they became popular on PCs. Their final hardware product was the MPlane, a low profile backplane for use in the MinisQL.

In addition to producing hardware TF Services was one of the first QL traders to set up a bulletin board and then, later, a website. To many QL users, especially those who had a simple use of the QL, TF Services will be remembered as a reliable repairer of their systems. One such user described how Tony Firshman had serviced 3 QLs, 2 monitors, 2 keyboards and a printer.

"Tony's charge was very reasonable for the amount of time he spent on my equipment. I will be 79 tomorrow and feel that I am now safe for the rest of my life."
In fact Tony's basis price for servicing a QL has remained unchanged for over 15 years.
At the moment Tony's time is heavily taken up with the rebuilding of a house in Leighton Buzzard and in his day job by Worldnews. He will probably only be able to get to QL shows in his near vicinity, but promises to continue the show email shots. He will also not be reordering components when his present stock runs out. This especially applies to the SuperHermes and Romdisq as they require large minimum orders of the main chips.

Other Trader News

QBRANCH

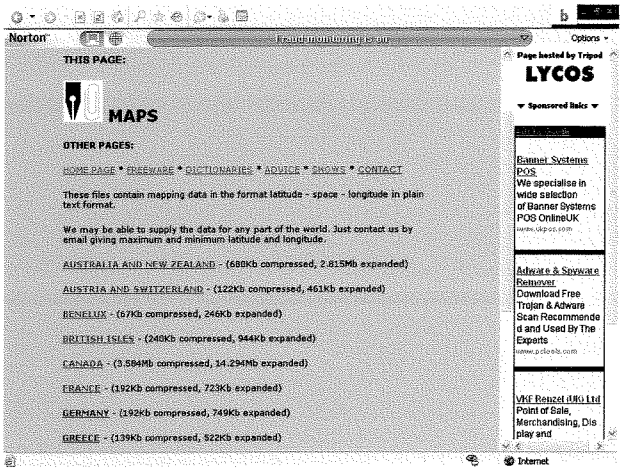
Roy Wood has closed QBranch's dedicated telephone and fax lines. He can now be contacted on:
TELEPHONE AND FAX: +44(0) 1273 430501
SKYPE: royqbranch
WEB: www.qbranch.demon.co.uk
Roy has recently been offering second hand QL hardware via the QL users email group.

J-M-S

Jochen Merz has improved his service to his overseas customers by opening bank accounts in several countries. He now has accounts in Germany, Austria, Switzerland, the Netherlands and the United Kingdom.
UK customers can pay by sterling cheque and USA customers in dollars. You can find the full details, including exchange rates, in the Jochen Merz Software advertisement.

New Web Page

Just Words! has added a maps page to its website. The new page has been launched to coincide with the mapping articles in this issue of QL Today. The mapping data is in the format latitude - space - longitude and can be loaded into Hugh Room's program without modification.
At the moment the page contains mapping data for most of the countries where QL Today readers live, but Just Words! is hoping to offer a "tailor made" service to give mapping data for any part of earth.
Unfortunately website bandwidth restrictions means that it is impossible to post the mapping data for complete continents on the web page, but Just Words! will be making this data available on CD for a nominal fee.
Just Words! has also made some minor changes to the site. The downloads page has been renamed "Freeware" and the position of links to other pages has been moved to minimise problems caused by the pop-up advertising.



SQLUG Update

George Gwilt informs us:
"For 2008 there are three new items on the Scottish Users site at:
http://jms1.supanet.com/

1. *SMSQ/E A simple quick method of compiling SMSQ/E v3.13 using the assembler GWASS*
2. *EXEG EXEG is a new keyword which is like EXEP in that it executes executable Things but, unlike EXEP it allows channel IDs as well as a parameter string to be put on the stack.*
3. *TURBO The latest version of TURBO, v5.05, allows two retry points, one for a WHEN_ERROR 0 clause and the other for WHEN_ERROR 1."*

Goodbye Netscape

Netscape, one of the most popular web browsers among QL-ers, is no longer being supported by the owners AOL and Time Warner. Their support ceased on 1st February this year. This means there will be no more upgrades.
Netscape was the original web browser and many of its supporters believe that without it the internet would not have come into being. However it slowly lost market share to Microsoft who integrated Internet Explorer into the Windows package. However the spirit and technology of Netscape will live on as its supporters have now developed the Firefox browser.

Last year 42% of people visiting the Just Words! website were Netscape users compared with 53% Internet Explorer and 5% Opera.

LEAR PCB CAD Update

Dilwyn Jones has informed us of a further update to this program:

"Version 6.39 of Malcolm Lear's PCB Cad program is now available from Dilwyn Jones's website. The most significant change is that you now have the option of running the program as a daughter job. The biggest advantage is a complete cleanup on termination which includes all functions and procedures loaded on program bootup."

The package is a 914KB download from:

http://www.dilwyn.uk6.net/graphics/index.html

QDT back on Track

Jim Hunkins resumed work on QDT at the beginning of February, which he had to suspend following an accident last year. In an email to QL Today he writes:

"Work on the FileManager for QDT has resumed, albeit slowly. My recovery from my severe accident last Fall has proceeded well enough that I can now spare some energy and time and therefore have resumed work on the project just this last week. No promises on dates as my paying job is even more time consuming that the previous year but the project will get completed."

No Go LINUX

At QL Today we do not know if Norman Dunbar got what he wanted from Santa, but we do know he was not granted his New Year wish. On the QL-users group he had a sad tale to tell:

"Windows ate my laptop recently. Over the festive period my XP system, fired up for the first time in absolutely ages, removed my Linux root drive (partition) without so much as a by-your-leave."

Norman decided to metaphorically throw Windows out of the window and set up a completely Linux laptop, but his troubles were not at an end:

"This of course, leaves me with a slight problem. When I go away, I usually take the laptop to watch DVDs, listen to my music, read 'books' etc, but also for QPC and my ongoing Assembly Language series. To this end, and I can almost hear Marcel groaning right now, I'm wondering what the possibility of getting a Linux version of QPC is?

Do I hear the work 'slim' or even 'no chance at all' out there?"

Marcel did indeed react using rather stronger language and outlined some of the problems involved:

"Well, step 1 would be to get the whole assembler stuff compiling under Linux, which given that Linux uses a completely different syntax for everything assembler (AT&T style versus Intel style) would be quite a feat in itself. Not to say practically impossible.

It might be more sensible to continue using a Windows assembler and hoping that the Linux linker can cope with the resulting object format. Not sure how well that would work out.

Portability wasn't really high up in the list (or even ON the list at all) of criteria for QPC's code and it shows. Due to historical reasons many parts are still mostly assembler, though over the years some parts migrated a bit to the C side of things. Most are a mix of both. All in all many things would probably have to be rewritten from scratch. If I rewrote QPC today it would be much different, but many of the design decisions were done when I was 15 or so, for much less powerful machines.

If you want to give it a try, that could probably be arranged. It would certainly be fun to have a Linux version, at least if I have nothing to do with it whatsoever. But quite frankly I think life is too short to even try."

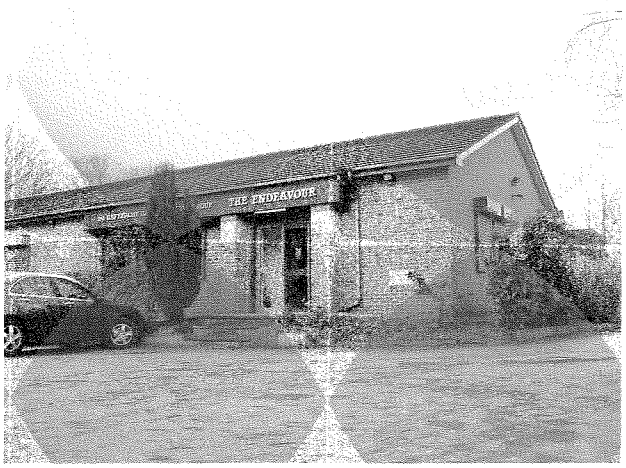
Last year Wolfgang Lenerz wrote an article for QL Today on using QPC under Linux (Vol. 11 issue 4 page 15), but this is not entirely satisfactory. Wolfgang used software called wine which allows some windows programs to run under Linux, but Norman was unable to get this working satisfactorily because of character repetition problems. Marcel commented that wine is a bit overwhelmed by QPC and perhaps Cedega would be better. However he thought the best performance could be achieved by running Windows in a virtual machine like VMWare.

There was a lengthy discussion on the QL-user group which did not provide a solution to Norman's problem. QL Today will be pleased to have an update to Wolfgang's article if and when the Linux specialists have made some progress.

QUANTA News

At the time of writing QL Today has not had details of the papers for the Quanta AGM

because these will not become available to members until after our deadline date. We hope to have a full news report in issue 4 as well as an account of the Manchester show and AGM.



However we understand that there is good news from Quanta who are anticipating some new committee members. We are unable to report further details in this issue because the Quanta committee strongly feels their members should be first informed of these via the Quanta Magazine. QL Today respects Quanta's wishes in this respect.

Meanwhile Quanta's Manchester subgroup have accidentally scored a home goal in their plans for the workshop and AGM in April. One of the big problems for Manchester workshops is hotel accommodation when Manchester United are playing at home, and the AGM is planned for the only weekend in April when Manchester United has a home game. To make matters worse the opponent is Arsenal, who are rivalling Manchester United for the top of the Premier League.

The Manchester subgroup has told QL Today that their hands were tied on the matter. The AGM could not be held earlier because the financial reports would not be available, and then the weekend of 12th and 13th April was the only time the scout hut was free. The subgroup is making active enquiries of a family hotel near the show venue to see if they can accommodate a Quanta party. A member who stayed at the hotel two years ago gave it favourable reports.

Website Update

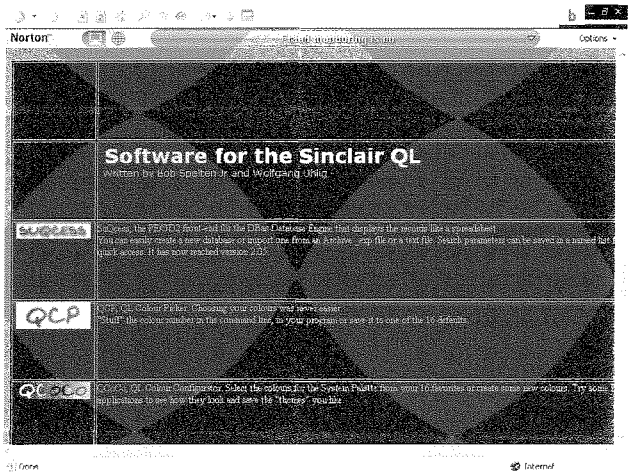
Bob Spelten has updated his website to include software In an email to QL Today he writes:

"Wolfgang Uhlig expressed his intention to close down the QL section of his website. That's why I made my own site at the launch of version 2.05 of our program SuQcess (see QL Today v12i1). I have now updated my site to include most of Wolfgang's software and some third party software from his site. It is all freeware apart from SuQcess of which a trial version can be downloaded. Wolfgang has also given me the source code of his programs and gave me permission to update them. I'm already working on new options for QCoCo."

To visit the site go to:

http://members.upc.nl/b.spelten/ql/

The home page of Bob Spelten's website contains links to download SuQcess, QCP and QCoCo. There is also a link to a utilities page of "Useful programming tools", which will be mainly of interest to QL-ers wanting to learnt about GD2 colours although there is also a utility for converting a DBAS file into a HTML file.



More Updates

The J-M-S update site

http://updates.j-m-s.com

features more updates of J-M-S products. Registered users can download many updates of J-M-S programs. The list of programs currently available is QSpread, MenuConfig, Menu Extension, QPAC1, QPAC2, EasyPTR V4, WinEd, CueShell, SuqCess, FiFi, BASIC Linker, SMSQ GoldCard (with and without Aurora drivers), SMSQ QXL, SMSQ ATARI (incl. mono), QD.

The updates of QPC and QPCPrint are not available from this site, as Marcel maintains them very well on his own site:

http://www.kilgus.net

The password to unzip QPC is the same as it was in the past, and to download the QPCPrint update, follow the instructions on Marcel's Website to log in (e.g. start QPCPrint and click on the "About" button to find out your license details required for the download login.

Assembling SMSQ/E with GWASS

by George Gwilt

The official source of SMSQ/E includes a large number of files containing assembler code. This source also contains a file explaining how to compile SMSQ/E. In it there is the remark:
For the time being, the assembler must be the QMAC Assembler.

This clearly suggests that at some time in the future other assemblers could be used. The reason that, at the moment, only QMAC can be used is that the source code is written in such a way that no other assembler would accept it. However, some years ago now, I produced for my own use a version of the source specifically for GWASS.
After a discussion with Marcel Kilgus and Wolfgang Lenerz at one of the QL meetings, I amended GWASS so that it would assemble code written for QMAC with little or no alteration. I also, with Wolfgang's knowledge, produced versions of the SMSQ/E source which could be assembled both by QMAC and by GWASS. If this code were put on the official site, then the current method of assembly by QMAC would be entirely unaltered. In addition, by using some programs which I can supply, the same source could be assembled by GWASS.
In order that such altered source code can safely be put on the official SMSQ/E site there must be acceptance that no errors have been introduced. Thus the binaries produced by QMAC from both the original and changed source must be identical, from a user's point of view. Also the binaries produced by GWASS should be effectively the same as the QMAC ones.

I have satisfied myself in both these cases. Indeed, GWASS versions of SMSQ/E from v3.12 have been used on the Q40, Q60, QXL and Super Gold Card without any detectable problems.

Wolfgang Lenerz has, reasonably enough, indicated that the altered source code could not be put on the official site unless he was sufficiently satisfied with the changes.

There are four ways I used to test the changes:
1. Compare the _ASM code by eye.
2. Compare the resulting assembled _REL files.
3. Compare the resulting linked modules.
4. Compare the resulting SMSQ/Es by use - rather like beta-testing.

In the hope that some SMSQ/E users can be persuaded to become beta-testers, I propose to put on the SQLUG site three files enabling users to compile their own versions of SMSQ/E by GWASS.

The three files consist of a zipped file containing necessary programs including GWASS, an SBASIC program and an explanatory text file.

The first part of the text file is:

"How to Compile SMSQE v 3.13 Using GWASS

REQUIREMENTS

1. Machine

QXL, Q40, Q60, Super Gold Card with SMSQE or QPC2 (v 3.33 or later) with at least 16 megabytes of RAM and at least 200 megabytes of space on the drive to contain the source code and answers.

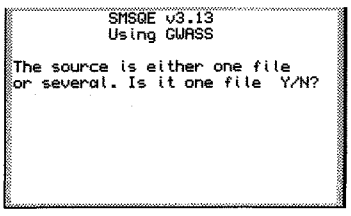
ACTION

A. Download the official source from Wolfgang Lenerz' website.
http://www.scp-paulet-lenerz.com/smsqe/

The source is available in two versions. Either in one large file (smsqe313.zip) or in several files (iod.zip etc). The first version is too large to fit on one HD disk.

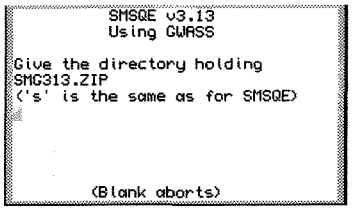
B. From our site download smg313.zip then save smsg3_bas and LRUN it on your QL."

When the program smsg3_bas is run a window appears into which some answers have to be set as is shown below.

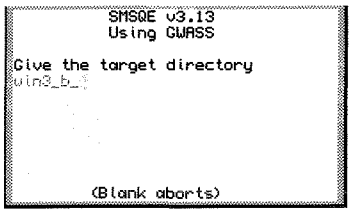


(asks if the source is one or many files)

(asks for the directory holding SMSQ/E source)

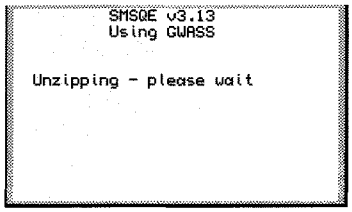


(asks for the directory holding SMG313.ZIP)

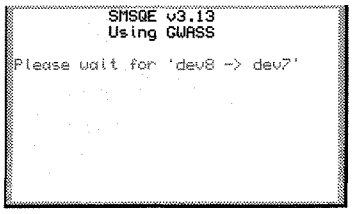


(asks for the target directory)

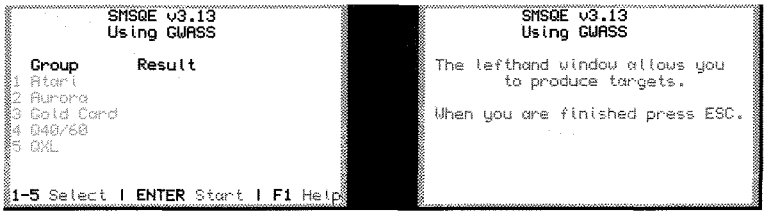
The target directory is the directory into which the source code will be loaded and in which the completed binaries will appear. This directory will contain 16 sub directories some of which will have their own sub directories. Since the resulting filenames can be quite long, it is important to keep the name of the target directory short, preferably restricting it to just one letter, as in the example.



(indicates that unzipping is occurring)



(indicates that dev8 -› dev7 is occurring)

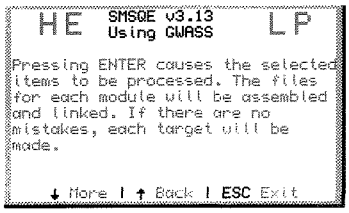


(a new window to allow choice of SMSQ/E)

The left hand window allows selection of the group of SMSQ/Es to be made.

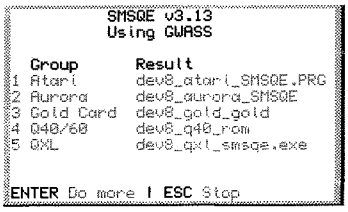
Further information can be found by pressing F1 to produce a help window as shown below.

(the 1st help window)



(the 2nd help window)

When the SMSQ/E group has been chosen the binaries are produced and set in the main directory as indicated below. This process takes around three minutes on a QPC2 to compile all five targets. On a Q60 it takes about ten times as long.



(shows a successful result)

If, at any stage, the left hand window becomes not available, by use perhaps of CTRL_C, it can be picked by pressing CTRL/SHIFT/ALT/DOWN.

Code Changes

I have made various changes to the SMSQ/E source code, many of them trivial. Trivial changes included moving "section" commands to the start and also putting a space between the end of an instruction and the start of a comment on the same line.

The most serious changes related to macros. The coding of macros is quite different in QMAC from that in GWASS, which followed the syntax of HISOFT and Talent's Thor MC68020/68881 Macro Assembler written by Eddy Yeung. This change required rewriting all the macros and setting them in a different sub directory from the QMAC macros.

Although some macros are called in the same way by both QMAC and GWASS most of them require different coding. To reduce the changes that would otherwise be required in the source code I altered GWASS so that there is now the option to process macro calls as for QMAC.

Final Comments

I have wondered whether or not the changes I have made to the SMSQ/E source comply with the conditions under which Tony Tebby made the source public. Put simply, the conditions appear to be that the source is made freely available but there should only be one official version of SMSQ/E at any time.
Since I have taken great pains to ensure that as far as is humanly possible any SMSQ/E version resulting from my altered source is indeed the same as the official version, I think that I have complied with the conditions.

It would be useful if anyone producing and using a GWASS version could let me know if there is any problem, either in its production or in its subsequent use.

Annotating Diagrams

by David Denham

Having just created a complex diagram using graphics commands like ARC, CIRCLE and LINE I had a need to add some text to the diagram and found it to be surprisingly difficult to accurately place text over the diagram.
Graphics commands like those use the graphics co-ordinate system, a scaleable system where the origin is at the bottom left of the co-ordinate system, like a graph (in contrast to commands like AT and BLOCK which down from the top of a window). The actual scale is set by the SCALE command:

```
SCALE#channel,scale_factor,x_value,y_value
```

The channel number is the window to which this scale command is applied, or window channel #1 if none is specified.
The scale_factor sets the number of units high the window is to be. For example, 100 sets the window to be 100 units high, independent of the height of the window in pixels. The width is then set in proportion to the ratio of window width to height and can be rather tricky to work out, but luckily the system is robust enough so that even if you try to draw somewhere off screen, the graphics commands merely work out what is on screen and plot those points and ignore anything which falls off screen.
If you set the window to be 100 units high, and then issue a command like CIRCLE 50,50,40, this draws a circle of 40 units radius, with its centre being at 50 units across and 50 units up from the default origin at the bottom left of the screen.

We can plot a single point at the circle's origin with the command POINT:

```
POINT 50,50
```

Suppose we wanted to place a "+" symbol at the centre of the circle. We could draw one using two lines, but we can also position text on graphics co-ordinates by using a variation of the CURSOR command with two extra parameters, which position the cursor by means of a graphics co-ordinate position, offset by a number of pixels indicated by the extra parameters:

```
CURSOR [#channel,] graphics_x,graphics_y,
x_offset,y_offset
```

(the channel number is optional, like the normal CURSOR command, defaulting to channel #1)

So, to place a "+" character at the centre of the circle, we specify the graphics co-ordinate (50,50) and an offset in pixels. Since we wish to centre the character, we tell it to put the "+" at the centre, less half the width of the character and half the height of the character:

```
CURSOR 50,50,-3,-5 : PRINT"+"
```

Important note: unfortunately, some versions of the QL ROM have a nasty bug which prevents the use of the five parameter version of the CURSOR command, but even on version AH and JM ROMs you can still get the command to work with 4 parameters, i.e. miss out the channel number, in which case it defaults to using channel #1. You can get more details of this bug by reading Simon Goodwin and Mark Knight's articles on QL bugs in past issues of QL World and QL Today respectively.
Suppose we wish to proceed to draw a clock face within the circle - this would require us to draw minute and hour points and numbers from 1 to 12 inside those number points.

We work from two more imaginary circles inside the clock face and use some fairly simple calculations to work out where to place the points initially. A full circle is 360 degrees, so the twelve hour points would be spaced by 360/12 or 30 degrees, while the 60 minute points would be separated by 360/60 or 6 degrees each.
Using this information we can create a simple program using just a few lines of basic to add the points and hour numbers to our clock face. The first loop, called minute, plots 60 equally spaced points on an imaginary circle 5 units inside the first (i.e. radius of 35). The second loop, called hour, plots twelve hour points on the same circle, all equally spaced, but in a different colour. Finally, we plot the numbers themselves on another imaginary circle inside the minute marks, using CURSOR to ensure that the centre of the digit falls in the right place, by subtracting half the pixel width and height at the end of the CURSOR command. I have used the default channel #1 for this to try to make sure this works on older QL ROM versions.

```
100 REMark Clock Face, by David Denham 2008
110 :
120 PAPER 2 : INK 7 : CLS
130 CIRCLE 50,50,40 : REMark outline
140 CURSOR 50,50,-3,-6 : PRINT "+";
150 :
160 REMark draw minute points
170 FOR minute = 1 TO 60
180   POINT 50+(35*SIN(RAD(6*minute))),50+(35*COS(RAD(6*minute)))
190 END FOR minute
200 :
210 REMark draw hour points in different colour
220 INK 0
230 FOR hour = 1 TO 12
240   POINT 50+(35*SIN(RAD(30*hour))),50+(35*COS(RAD(30*hour)))
250 END FOR hour
260 :
270 INK 7 : REMark back to white
280 :
290 REMark plot the digits
300 FOR hour = 1 TO 12
310   CURSOR 50+(30*SIN(RAD(30*hour))),50+(30*COS(RAD(30*hour))),-3-(3*(hour>9)),-5
320   PRINT hour;
330 END FOR hour
```

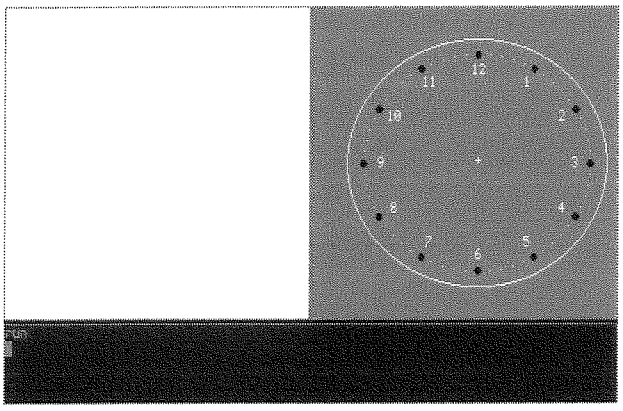


Figure 1: Output of the clock display program

The program draws the clock face shown in Figure 1. It should now be a fairly straightforward job to plot the minute and hour hands using the angles shown above for minutes and hours (and seconds if you wish) and another imaginary circle just inside the digits. Using OVER -1 and drawing the hour, minute and second hands twice as they change, this will undraw each hand as it is time to move it. Refer back to my time and clocks articles ("Clocking In") in past issues of QL Today if you wish for further information.
If you wish to make the hour dots more visible, one way would be to draw a filled circle in line 240, rather than just use a POINT:

```
240 FILL 1: CIRCLE 50+(35*SIN(RAD(30*hour)
)),50+(35*COS(RAD(30*hour))),1 :FILL 0
```

Another fairly obvious use for CURSOR is to annotate graphs. The graph may be drawn by a series of POINT and LINE graphics, and text positioned at a given point by calculating where the point is on the graphics co-ordinate system and using CURSOR to map the text onto the graph, possibly with a suitable number of pixels offset to make sure that the text doesn't overwrite some of the graphics:

```
100 REMark Graph example
110 PAPER 2 : INK 7 : CLS
120 SCALE 100,0,0
130 LINE 10,90 TO 10,10 TO 90,10
140 :
150 FOR y = 0 TO 9
160   CURSOR 10,10+8*y,-12,-10 : PRINT y
170 END FOR y
180 :
190 FOR x = 0 TO 9
200   CURSOR 10+8*x,10,0,5 : PRINT x
210 END FOR x
220 :
230 REMark draw a random 'graph', labelling the random points
240 INK 4 : LINE 10,10
250 y1 = 10 : OVER 1
260 FOR x = 1 TO 9
270   y1 = y1+RND(1 TO 10)
280   LINE TO 8*x+10,y1
290   CURSOR 8*x+10,y1,1,1 : PRINT CHR$(64+x)
300 END FOR x
310 INK 7
320 CURSOR 10,90,6,-10 : PRINT 'RANDOM NUMBER GRAPH'
```

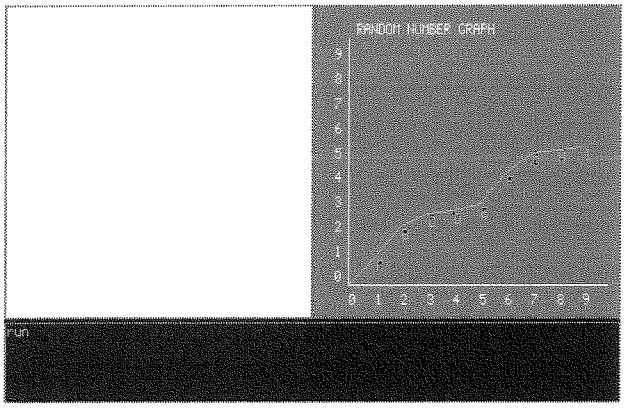


Figure 2: Output of the graph labelling program

Figures 1 and 2 respectively show the output you should expect from the two examples. Work through them and by altering the text and graphics you can experiment a little to gain more insight into how the command works.

You will have seen by now that what the 4 or 5 parameter version of CURSOR does is to position the text cursor position at a given point using the graphics co-ordinates, allowing you to synchronise text and graphics positions with relative ease, making it easier to position text on graphics which use the graphics co-ordinate system. There are problems getting this version of CURSOR to work on some older QL ROM versions, but if you are able to use screen channel #1 or are using a more recent ROM version this can be a handy addition to your graphics programming knowledge.

QL2K

by Dilwyn Jones

David Denham wrote about the QLay2 emulator for Windows machines in issue 2. In the same issue, Jimmy Montesinos gave us news of the latest version of the QL2K emulator.

QLay2 was a straightforward and free development of the original QLay emulator.

QL2K takes the emulator into a new direction. This version is now described as Postcardware or Registerware. Basically, it is also a free QL emulator, but if you use it, you should register with the author so that you can receive news about updates, be a part of a QL2K "community" and so on.

QL2K is intended for use on newer versions of Windows: Windows 2000 and Windows XP for example. The author suggests that it has been successfully tested on Windows Me (Millennium edition) too.

The QL2K emulator is described as still being in the alpha-release stage of development. Version 0.98a is the latest version available at the time of writing - you have to download version 0.96a from

http://www.jadiam.org/QL

and then download the version 0.98a patch to update it. 0.96 is a download of about 467K, while the 0.98a update is about 180K in size. You have to run the setup program for v0.96a, then apply the patch to update to 0.98a

Once you have installed all the files, the emulator is started using a shortcut to QL2K.EXE and the configuration screen (see figure 1) pops up. Here you can select a language for the prompts by clicking on one of the flags at the bottom, memory size (from 128K, 640K, 1M, 2M, 4M or 8M), the display size (always QL 512x256, but this can be mapped onto any of 8 PC resolutions to allow it to look bigger on high resolution PC displays), Keyboard country, Clock Multiplier, Fast Microdrives, any of up to 6 add-on ROMs, use DirectDraw, Full Screen GDI, No mouse, and Autostart. An additional button at the bottom leads you to a second configuration screen where you can assign path names to up to 8 WIN and 8 MDV drives.

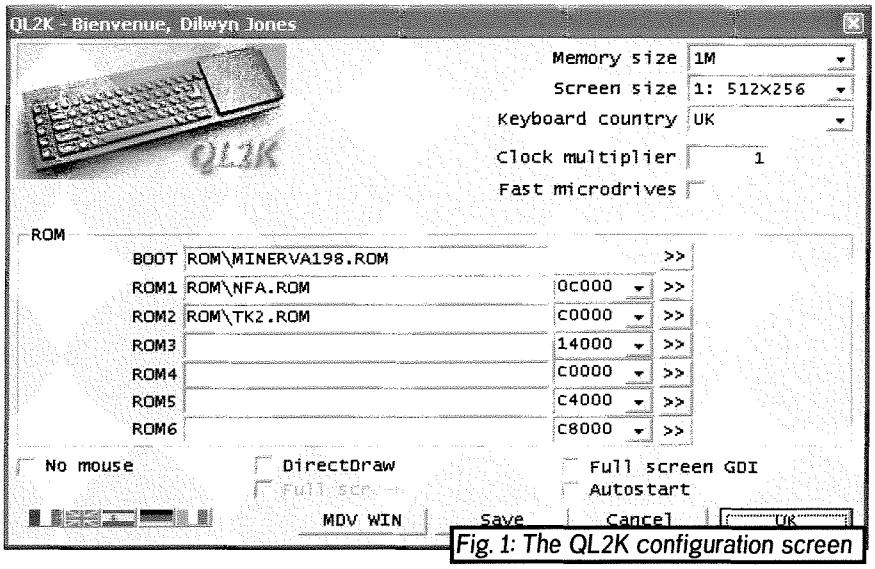


Fig. 1: The QL2K configuration screen

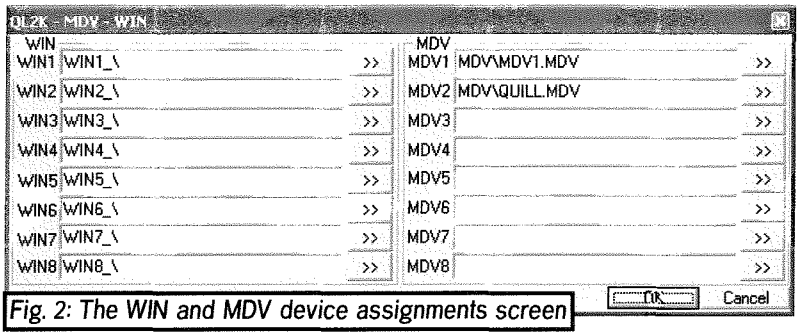


Fig. 2: The WIN and MDV device assignments screen

The main difference between this and QLay 2 is in the field of execution timing. You should find that QL2K does not "slug" your PC any more - the problems David reported don't seem to exist on QL2K (slow screen update, hogging PC resources). Now, a QL set at original QL speed uses very few percent of CPU time in both DirectX and what it calls "GDI mode". There is a clock multiplier function, so if you are running QL2K on a reasonably fast PC, it can be set to run at, say, twice the speed of a QL. This seems to be a true speedup too, which doesn't seem to speed up a PAUSE delay for example, as older versions of QLay used to tend to do when coaxed into speeding up. Certainly, entering something like

```
d=DATE:PAUSE 3000:PRINT date-d
```

prints the correct value of 60 seconds. You can set the speedup in the configuration screen (Clock Multiplier) or via a command line switch, where the command to start QL2K.EXE has the switch `-P x` appended (where x is 1 for original QL speed, 2 for twice normal speed and so on). This is very useful - games can be run at original QL speed to prevent them running too fast to be playable, whereas you can make the emulator run faster for time consuming tasks.

There is now an option to set the microdrive delay from 31ms down to 1ms. As I don't use the MDV emulations I can't comment on this.

A debugger is now available from the COMMANDS menu. I haven't tried to use, other than to start it and quit from it using ESC.

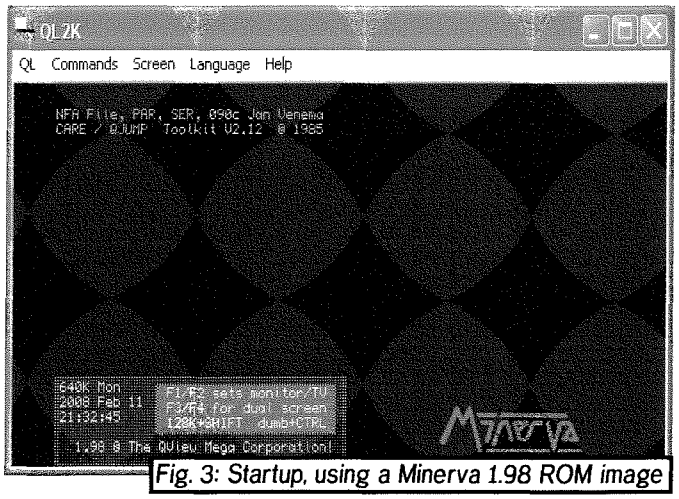


Fig. 3: Startup, using a Minerva 1.98 ROM image

One of the things which still doesn't work is sound. Enter a BEEP command and nothing happens - the command is ignored. And there's still no floppy disk driver - you still have to use the supplied tools programs to copy files to or

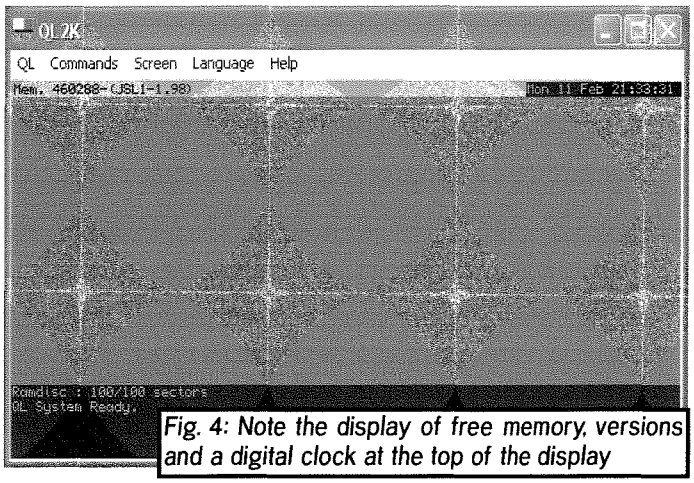


Fig. 4: Note the display of free memory, versions and a digital clock at the top of the display

Compatibility

In summary, this is a very worthwhile progression to the earlier QLay emulators. The fact that the emulator doesn't slug the PC any more is great - I can now have the TV tuner running on the PC at the same time as QLing on QL2K without one or the other struggling. The 'clock multiplier' facility is very useful too, as it lets you choose between running older games at standard QL speed or taking advantage of the higher processing speeds possible on modern PCs.

from QL disks. The available 8 WIN drives should mean that this won't be too big a problem, as you will be using the WIN drives on your PC's hard disk most of the time, only resorting to QL floppy disks if you wish to copy files between the emulator and another machine. It is still necessary to fiddle with the Tools programs to update the WIN directories if files are copied to or from the PC side of things.

There is a Help command available, but it needs an active internet connection to take you to online QL2K documentation at the Jadiam website. You can download some documentation and a personal copy of Simon Goodwin's Speedscreen+ ROM for use with QL2K (see figure 6).

The requirements that QL2K makes of your PC are fairly modest - minimum of Windows 2000 (although claimed to work with the older Windows Me), 9MB free RAM, 3MB hard disk space and DirectX 8.0 or later.

The startup screen ROM titles claim that PAR and SER is emulated in the NFA ROM title. As my printer is connected to my PC via USB, I can't use my printer it would seem, though it may be possible to use a parallel printer.

QL2K can use most versions of the QL ROM, including Minerva, so if you have an awkward program which only works on one particular QL ROM, it is quite easy to boot the emulator with a copy of that ROM image to run the program in question.

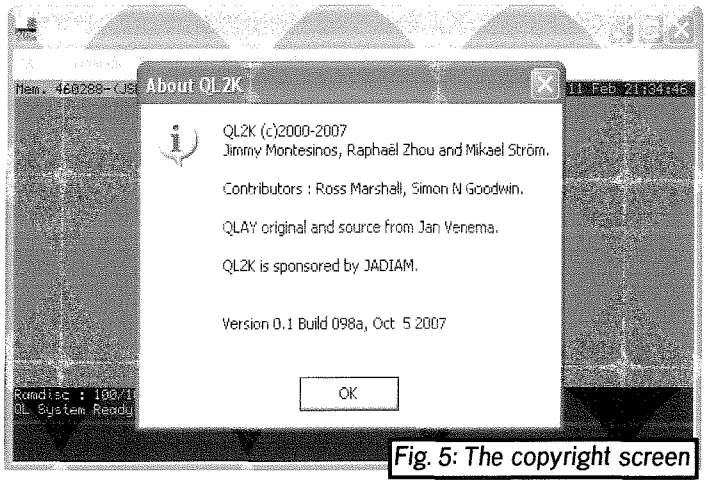


Fig. 5: The copyright screen

QL2K is a completely free emulator at the moment, apart from the need to register with the author Jimmy Montesinos (registration is free - just fill in a form on his website).

www.jadiam.org/QL

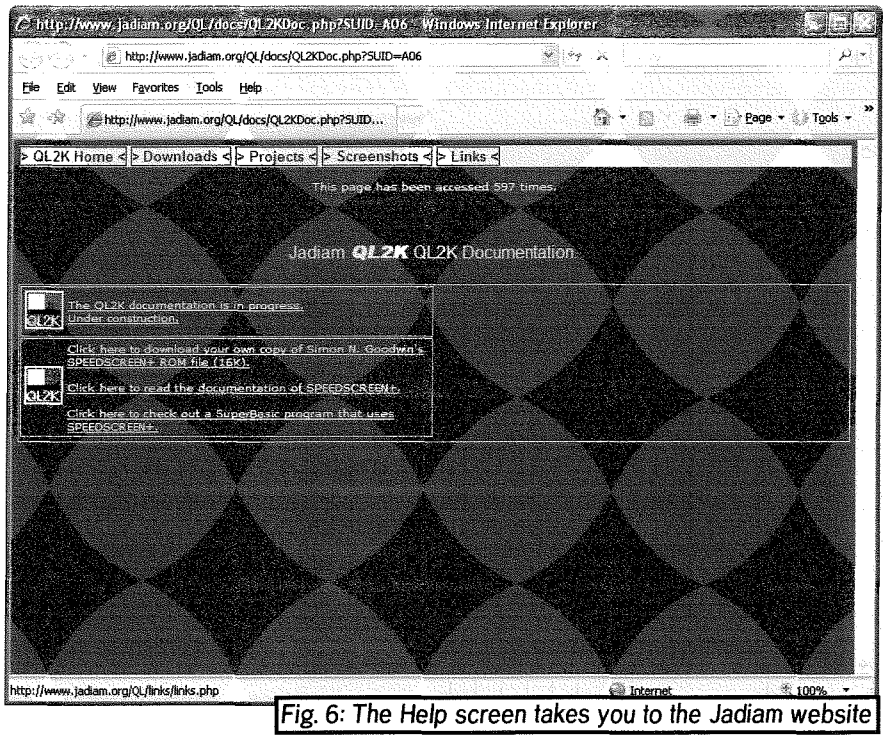


Fig. 6: The Help screen takes you to the Jadiam website

Programming in Assembler - Part 20
The Pointer Environment

by Norman Dunbar

Well for many years now I've been avoiding this moment, but it has finally arrived. I am starting to learn all about programming the PE in assembly language. You are coming along for the ride! I'm sure that along the way I'll be making as many, if not more, errors that I usually do and someone out there who knows far more than me, will correct me as we go along. As I always say, you should learn from your mistakes!

There are two parts to the PE, ptr_gen and wman - the Pointer environment and the Window Manager. To slip easily into programming, we shall start off with a small and perfectly useless pointer only program, so the Window Manager stuff will not be used.

In order to run the program you should have PTR_GEN loaded. If you are running SMSQ/E then it is built in to the operating system. If you are running a 'normal' QL then you will need to load it. I suspect most - if not all - users still with us will have a copy. However, if not, Dilwyn's Web Site will have a copy to download.

Ok, lets dive straight in with our next to useless program, beginning with a host of equates.

```
me          equ     -1      ; The current job id
timeout     equ     -1      ; Infinite timeout
openOld     equ     0       ; Open Old Exclusive device
iop_pinf    equ     $70     ; Get PE information
iop_outl    equ     $7a     ; Outline a primary window
iop_rptr    equ     $71     ; Read the pointer
termVec     equ     $01     ; When to stop reading the pointer
```

The final equate above tells the call to IOP_RPTR when to return back to the code in our program. It is 8 bits long and each bit has special meaning, as follows :

Bit	Meaning
0	Return when a key or button is pressed in the window. Also, request window resize.
1	Return when a key or button is pressed (subject to auto repeat). Also request window move.
2	Return when a key or button is released in the window.
3	Return when the pointer moves from the given co-ordinates in the window.
4	Return when the pointer moves out of the window.
5	Return when the pointer is inside the window.
6	Reserved. Do not set.
7	Window Request.

Bits 0 and 1 look a bit funny and are used in conjunction with bit 7 to indicate a window request. If bit 7 is set then the rest should be zero except bit 0 or 1 - one of these should be set. If bit 0 is set the pointer is the resize one and if bit 1 is set it is the window move pointer. If both are unset, the Window REquired pointer will be displayed. With bit 7, the topmost window of the pile if hit and selected. More on this later in the series.

Our termination vector is set to have bit zero turned on only. This means when a mouse button is pressed or any key is pressed while the pointer is in the window we create later, the call to IOP_RPTR will return.

As this is a job, we need a standard job header and as we have seen this so many times before, I shall not insult your intelligence by explaining it yet again!

```
    bra.s  start
    dc.l   0
    dc.w   $4afb
```

```
        dc.w  15
        dc.b  'PTR_GEN Test v1'
```

Next we have a few definitions of channel names, window sizes and space for the Pointer Record that we get back from a call to IOP_RPTR. First of all, a channel to a console is required.

```
conChan     dc.w    4
            dc.b    'con_'
```

Once we have it open, we redefine it to be 200 wide, 100 deep and centred in the middle of a 512 by 256 'standard' window using the following block of 4 words.

```
conDef      dc.w    200,100,156,78
```

The next 24 bytes are used by the IOP_RPTR call and it stores the pointer record. There is more on this later on in the text.

```
ptrRec      ds.w    12
```

This is where we start the main code. It's pretty simple as you can see and all we basically do here is open a console device, redefine it as mentioned above, set a border of one pixel in red and finally clear the screen (to black paper by default) If anything gives an error we simply kill the job and exit.

```
start       moveq   #1,d0         ; Strangely, using #io_open kept giving compile errors!
            moveq   #me,d1        ; Channel is required for this job
            moveq   #openOld,d3   ; Device exists
            lea     conChan,a0    ; String defining device to open
            trap    #2            ; Do it
            tst.l   d0            ; Ok?
            bne.s   exit          ; No, exit

            moveq   #sd_wdef,d0   ; Redefine open window
            moveq   #2,d1         ; Border colour is red
            moveq   #1,d2         ; Border width is one pixel
            lea     conDef,a1     ; Console definition block
            trap    #3            ; Do it
            tst.l   d0            ; Ok?
            bne.s   exit          ; NO, exit

            moveq   #sd_clear,d0  ; CLS
            moveq   #timeout,d3   ; Infinite timeout
            trap    #3            ; Do it
            tst.l   d0            ; Ok
            bne.s   exit          ; No, exit
```

That's about all the main setting up that we have to do. We now have a channel open redefined and a nice border showing. The next stage is to look for the PE, if it isn't found, we have a problem and simply exit.

```
FindPE      moveq   #iop_pinf,d0 ; Get PE Information
            moveq   #timeout,d3  ; Infinite timeout
            trap    #3           ; Do it
            tst.l   d0           ; Ok?
            bne.s   exit         ; No, exit
```

So far so good. If the PE exists, we now need to make sure that our window is outlined. This indicates to the PE that the window is to be 'managed' It also defines the limits of the 'hit area' where a hit or do or keypress will be registered by our program. This gets a better explanation later in the series.

```
GotPE       moveq   #iop_outl,d0 ; OUTLN our window
            move.l  #$00210002,d1 ; Set a shadow size of 2 length and width
            moveq   #0,d2        ; Do not preserve the window contents
            moveq   #timeout,d3  ; Infinite timeout
            lea     conDef,a1    ; The size of the window NOT COUNTING the shadow size
            trap    #3           ; Do it
            tst.l   d0           ; Ok?
            bne.s   exit         ; No, exit
```

The shadow size is added to the sized defined in our console definition block but the shadow is outside of the hit area for our window. Now we read the pointer. The call to IOP_RPTR will not return unless the timeout expires or an event happens that has been set in the termination vector to cause a return. We are looking for a button or keypress while the pointer is inside our window.

```
Pointer     moveq   #iop_rptr,d0 ; Read the Pointer
            moveq   #0,d1        ; Initial pointer co-ordinates X,Y
            moveq   #termVec,d2  ; When to return - on button or keypress
            moveq   #timeout,d3  ; Infinite timeout
            lea     ptrRec,a1    ; Storage for 24 byte pointer record
            trap    #3           ; Do it
            moveq   #0,d0        ; We ignore CHANNEL NOT OPEN errors
```

When the user clicks in the window with the mouse, either button will do, or presses a key, the call to IOP_RPTR will return having filled the pointer record with useful information. We are not bothering with that here in this first simple demo. We are also ignoring the possibility of A0 pointing at a channel that is closed because by the time we get here, we have carried out lots of actions on it - so it should still be open!
The next part of the code simply kills off our job reclaiming any resources it allocated and closing channels etc.

```
exit        move.l  d0,d3        ; Save any error codes
            moveq   #5,d0        ; MT_FRJOB also gives compile errors!
            moveq   #timeout,d1  ; The job to kill is this one
            trap    #1           ; Kill me
            bra.s   exit         ; We should never get here, but just in case
```

That is all there is to this small demonstration. When assembled and run, you should see a 200 by 150 window centred on screen (well on a standard QL screen anyway) cleared to show black paper with a single pixel red border and a shadow down the right and bottom borders. It is waiting for you to move the pointer into. When you do it will change to an arrow and will remain as an arrow while you move it around. Click a button or press a key while the pointer is in our window and the job will kill itself.

So there we are, a very simple pointer Environment program. More next time as we extend our programming knowledge of the PE. See you then.

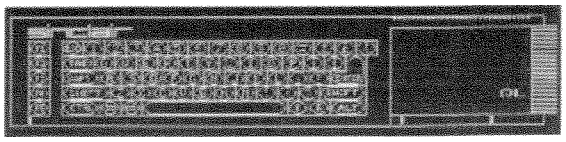
A Tick Timer

by George Gwilt

If you want to find how long a process or a program takes you can use the keyword DATE which is a function returning the number of seconds since the beginning of 1961. The increase in the value of DATE will give you the elapsed time in seconds. You can also find this time inside an assembler program or even a C program by using the trap #1 call MT_RCLCK which is in fact what DATE uses.

However, a second is quite a long time in computing. I have recently needed to measure durations in units of ticks rather than seconds. A tick is a 50th or 60th of a second, depending on the nationality of the QL. For the UK a tick is a 50th of a second.
Simon Goodwin produced TIMING_CODE as part of the DIY KIT reported in QL World May 1989.

QUANTA



Independent QL Users Group

World-wide Membership is by subscription only,
Offering the following benefits:
Bimonthly Newsletter - up to 40 pages
Massive Software Library - All Free!
Free Helpline and Workshops
Regional Sub-groups. One near you?
Advice on Software and Hardware problems
Subscription just £14 for UK members
Overseas subscription £17

Barclaycard: Visa: Access: MasterCard: Accepted

Now in our Twenty Fifth Year

Further details from the Membership Secretary

John Gilpin, 181, Urmston Lane
Stretford, Manchester, M32 9EH (UK).
Tel. +44 (0) 161 865 2872
Or
Visit the Quanta Web Site
http://www.quanta.org.uk
Email: membership@quanta.org.uk

Next QUANTA Sponsored Event

NEMQLUG
MANCHESTER WORKSHOP & QUANTA AGM 2008
AT

3rd Davyhulme Scout Headquarters,
"The Endeavour", Conway Road,
Davyhulme, Manchester. M41 0TE

Saturday & Sunday 12th/13th April 2008
From 10.00 am. To 4.00 pm. Daily. AGM at 2.00 pm. Sunday.
Ring Sarah Gilpin on
0161 - 865 2872 for full details.

This introduces six keywords, T_START, T_STOP, T_RESTART, T_ON, T_OFF and T_COUNT. These allow the use of up to four stop watches all counting in ticks. However, they can only be used in SuperBASIC. Since I wanted to be able to make my time measurements using assembler and C I had to think up my own system.

Tick Counter

Qdos is set up so that a 68000 level 2 interrupt occurs each tick. Qdos also allows a user to add his own routine to what is called the polled list. This is the list of routines which are called at each tick. All I needed to do then was to add a routine to that list. This routine would add 1 to a counter.

What is less easy is to decide where exactly this counter should be. There are two places accessible by all programs – the system variables themselves and the set of Things. Since Things are not available to all QLs it seemed preferable to locate the counter in the system variables. Normally this would be a very bad move since any free space in this area is in danger of being appropriated by any programmer who wants it for his own purposes. Thus normally I would never suggest using an area in system variables. However, the four long words at sys_vars + $D0 are officially set aside for the saving and restoring of the Floating Point Unit (FPU) when there is one, as in Q40 Q60 and some QXLs. Since I wrote the necessary software myself, I know that the fourth long word is not needed for the FPU save and restore. Thus I proposed to use this for the tick counter.

Linking to the Polled List

The following short program can be LRESPRd to set the tick timer going as part of the polled list. Each link consists of two long words. The first of these is used by MT_LPOLL to point to the next link. The second points to the routine to be used.

```
        lea     lnk,a0          address of link
        lea     rout,a1         address of routine to be linked
        move.l  a1,4(a0)        set routine's address in the link
        moveq   #mt_lpoll,d0    link . .
        trap    #1              . . it in
        moveq   #0,d0           return to . .
        rts                     BASIC

lnk     ds.l    2               link
rout    addq.l  #1,$DC(a6)      add 1 to the tick counter
        rts
```

Second and Third Thoughts

The small program above has two drawbacks. The first, potentially dangerous, is that it relies on LRESPR for its operation. One essential property of a link to the polled list is that it must not disappear. Its space from the heap must be owned by master BASIC. So, if you LRESPR the program from a daughter BASIC which is subsequently removed the machine will almost certainly crash.

The second drawback is not fatal but it can spoil the timing. There is nothing to stop the program being LRESPRd more than once. Each time this happens a new link is entered into the polled list with the result that the count goes up at each tick by the number of tick timer links.

A Safe Link

I thought the best way to set up a safe link was to do this through an executable program which would create the link in heap space having master BASIC as owner. This program could be executed from any BASIC or even from other executable programs.

Another, though less important, reason for having an executable program is that the space occupied by the link and the associated linked routine need not contain the testing and linking instructions and so will take up less of the heap. The linking program itself would gracefully retire after performing the link.

Checking the Tick Timer

For my own machine I would know that the timer had not started if the count were zero, especially if I arranged to set the count at 1 when I set the link. However, for other machines where there was the possibility of unauthorised use of the system variables a stricter check might be needed. As a compromise, if I find that the count is not zero I wait for a short time and then examine

the count again. If it had changed I assume the timer is working. Otherwise I know that the timer has not been set and also that someone else is using the space. In that case it is wise to exit leaving a message that the timer can't operate.

Error Messages

I have just indicated that the linking program should leave a message, at least if there is some fault. I decided to leave a message whatever the outcome. The messages would include something on the lines of:

1. Timer loaded OK.
2. Timer already loaded.
3. Timer can't be loaded.

Assembler and C

I then thought that it might be useful if the linking program could be called by an assembler program or by a C program. In these cases it would not be sensible to have windows popping up when the timer was set, or not set, as the case might be. Instead, I thought it would be better if the linking program could send the error messages as an ordinary error code.

The mechanism for this is as follows. The linking program when it removes itself by MT_FRJOB sets the error code in D3.L. If the linking program was called by EW or its equivalent, rather than by EX, the waiting program receives the error message in D0.L. If the calling program was BASIC the error message corresponding to the code would be printed in #0.

Config Block

My next thought was that if I had to choose between "window" and "message" for the departing remarks I should have a config block allowing the user to choose which method he wanted. I have always found the setting up of config blocks slightly daunting. For a long time now I have used a program T_CONFIG_DATA to produce specialised blocks for programs to be compiled by Turbo. Much more recently I have produced a generalised version of this program which is called UCONFIG (for Universal CONFIG). At each run this produces config blocks for BASIC, assembler and C.

Thus I found it easy, (at last!), to produce the assembler program's config block allowing the choice needed.

Parameter String

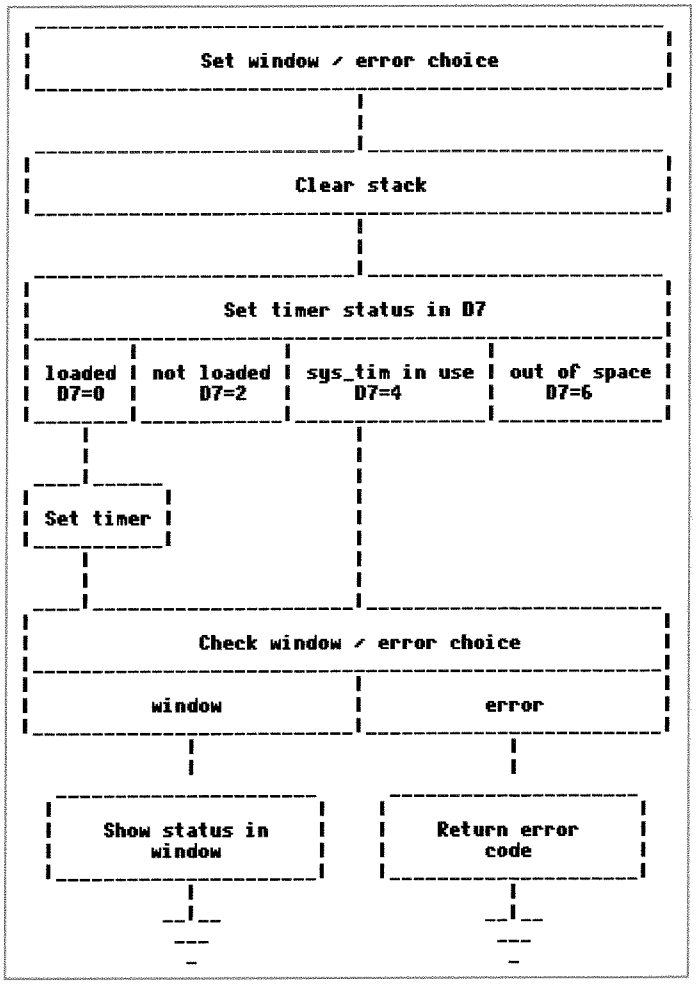
I have never found it reasonable to allow a choice via a config block without also allowing the same choice by a parameter string. The parameter string will, naturally, be expected to override the config block's setting.

Thus a string of three bytes is allowed. The first two must be "-w", with the "w" lower case. The third byte must be either "0", for "window" or "1" for message.

A Flowchart

A simple flowchart showing the present state of the linking program is shown here. As you can see, it is rather longer than the original LRESPR program which was, effectively, only the block called "Set timer".

"sys_tim" in this flowchart stands for "$DC", which is the place in the system variables used for the counter.



I hope in a future article to show a listing of the whole program with more detailed comments.

Random Access Files

by Dilwyn Jones

In this article I hope to show how to set up and use data files on disk so that you can create your own database and general data files.
Data files are created using the OPEN_NEW command or similar in SuperBASIC. Once you have opened the file, commands such as PRINT and PUT can send the data in an ordered fashion to the file, allowing it to be read back later.
The simplest type of data file is a simple list of names or words. In SuperBASIC we can simply hold the information in an array and just use PRINT to send it all to a file one after the other, then use the CLOSE command to finish it all off.
Here's a very simple example, where we dimension an array to hold up to 10 entries, and then the program opens a file, prints the elements of the array to the file and closes it, so that it is stored for later use.

GLOSSARY OF TERMS

DATABASE - A file which holds information as a list of entries

RECORD - One complete entry in database, equivalent to one page or one entry in an address book.

FIELD - One part of a record, e.g. a street name in an address list

FILE POINTER - a value which indicates how far into the file we are, or where the next piece of data will be read from or written to.

ARRAY - A SuperBASIC list, which can store strings, floating point numbers, or integer numbers, depending on what type it was declared as in a DIM statement.

CHANNEL - A number defined in an OPEN, OPEN_IN, or OPEN_NEW statement which indicates which file or device the data is to be printed to or inputted from.

```
100 DIM entry$(9,20)
110 CLS
120 FOR a = 0 TO 9
130   INPUT "Entry ";(a);": ";entry$(a)
140 END FOR a
150 INPUT"Filename ";filename$
160 OPEN_NEW #3,filename$
170 FOR a = 0 TO 9 : PRINT #3,entry$(a)
180 CLOSE #3
```

If you now copy the file to the screen, you can see that the data file looks like any old text file - each entry is stored to all intents and purposes like a line of text. For example, if you entered numbers when invited you should get a list which looks like this:

one
two
three
four
five
six
seven
eight
nine
ten

There are a few points of interest to look at in the above program.

On the QL, arrays dimensioned include one element of subscript 0, so DIM entry$(9,20) actually creates 10 strings (0 to 9) with a possible length of up to 20 characters. If you try to enter something longer than the value dimensioned (20) it is truncated to the first 20 characters. This is because the DIM command sets aside space for the amount of data specified and once it is set you can't change this maximum space without using another DIM command which of course destroys the original data and sets up a new definition. This is something we will need to consider when loading the information at a later date - the array should be dimensioned to match those which were used to create the file in the first place.
Note, how in line 130, the variable a is enclosed in brackets. This turns it into an expression, so that INPUT prints the value of the variable rather than inputting it.

Line 170 prints all 10 elements of the array to the file. An alternative to this is to change line 170 to print the entire array in one go using the much simpler form 170 PRINT #3,entry$ - SuperBASIC recognises that "entry$" is an array, and no subscript or slice has been specified, so it prints the whole array with a linefeed between each part of the array. The slight advantage of using a FOR loop to do the printing though is that it is a little bit clearer to read.
If we want to read the array back in later, we need to use an OPEN_IN command to open a channel to the file and then use INPUT to read the values back into the array, like this:

```
200 CLS : INPUT "Filename ";filename$
210 DIM entry$(9,20)
220 OPEN_IN #3,filename$
230 FOR a = 0 TO 9 : INPUT #3,entry$(a)
240 CLOSE #3
```

One way of improving our simple little file program is to ensure that we add data to the file which describes the dimensions of the array, to make sure we use exactly the same array when we load the file at a later date. To do this, we add two extra variables with the two dimensions of the array, and save these at the start of the file to allow this information to be recovered before the data itself is read back.
So our program to create the file now looks like this:

```
100 INPUT "Maximum length of each item ";wide
110 INPUT "How many entries ";high
120 DIM entry$(high-1,wide)
130 CLS
140 FOR a = 0 TO high-1
150   INPUT "Entry ";(a);": ";entry$(a)
160 END FOR a
170 INPUT"Filename ";filename$
180 OPEN_NEW #3,filename$
190 PRINT #3,wide
200 PRINT #3,high
210 FOR a = 0 TO 9 : PRINT #3,entry$(a)
220 CLOSE #3
```

And the program to read data back has to be modified to recover and use this information:

```
300 CLS : INPUT "Filename ";filename$
310 OPEN_IN #3,filename$
320 INPUT #3,wide
330 INPUT #3,high
340 DIM entry$(high-1,wide)
350 FOR a = 0 TO high-1 : INPUT #3,entry$(a)
360 CLOSE #3
```

What we have discussed thus far describes the very simplest type of file, called a SERIAL data file, so called because when loading or saving it, we always start from the beginning and work our way through to the end. This is a very simple type of data file to use, and is perfectly adequate for small, simple files.

The other type of data file is called a RANDOM ACCESS type. This is rather more complex than the SERIAL type, and as its name implies you can grab any piece of data from somewhere in the middle of the file and just manipulate individual items of data without having to load it all into memory. This allows very large files to be handled, but the programming involved can be much more complex.
The basis of a random access file is that all the entries in the file are of equal length or padded out to be equal length. This in turn lets us just work out how far into a file the data should be, position the file pointer there and we can fetch one item out of the middle of the file or write an item there to update the old value.

To visualise how a random access file would look, we'll use our example array above, and pad it out with spaces. I'll use a dot here to represent a space. Each line would end with a linefeed (shown by ‹LF› below). Our array "entry$" just contains the words "one", "two" and so on up to "ten". Our file would look like this:

```
one.................‹LF›
two.................‹LF›
three...............‹LF›
four................‹LF›
five................‹LF›
six.................‹LF›
seven...............‹LF›
eight...............‹LF›
nine................‹LF›
ten.................‹LF›
```

We dimension the array to be up to 20 characters wide. When we print it to the file, we make sure that exactly 20 characters are sent to the file and if necessary we add some spaces to pad it all out to the required length. Therefore, each entry becomes exactly 20 characters wide plus the linefeed which PRINT adds, so a total of 21 characters per entry.

```
100 CLS : INPUT"Filename ";filename$
110 DIM entry$(9,20)
120 FOR a = 0 to 9
130   INPUT "Entry number ";(a);": ";entry$(a)
140 END FOR a
150 OPEN_NEW #3,filename$
160 FOR a = 0 TO 9
170   PRINT #3,entry$(a);FILL$(' ',20-LEN(entry$(a)))
180 END FOR a
190 CLOSE #3
```

The important line here is line 170. This ensures that enough spaces are added by a FILL$ command to ensure that each entry in the file is exactly 20 bytes long plus the linefeed after it.
This has now created a file where we can work out where in the file each entry lies. This lets me introduce the concept of the file pointer, which is an operating system pointer which tells us how far into the file we are at the moment, in other words where the next entry will be taken from. It starts at 0, which means the beginning of the file (or how many bytes of data there are from the beginning of the file to here). In this case, we know that each entry is 20 bytes long and there is a linefeed after each one, so the first entry starts at 0, the next one at 0 plus 21, then next one at 0 plus 21 plus 21 and so on. So to skip the first entry we point just after the first 21 bytes. This first entry occupies bytes 0 to 20, so we set the pointer to 21 and INPUT the element starting there. So, to specify this mathematically we need to point to 21 x entry_number (where entry number starts from 0 like an array subscript). To get the third record we would set the file pointer to 42 (or 2*21) and so on.

Records and Fields

At this point we'll digress to discuss some terminology - RECORDS and FIELDS.

A Record is one complete entry in a database. A Field is one part of the complete entry. Suppose we are building a database of names and addresses. A record is one particular name and address. The lines within the name and address are each a different field:

Fred Bloggs
123 The Street
Anytown
Any County
England
AB12 3CD

If we use a different array for each line of the name and address, each array is a field, for example:

```
DIM name$(9,20)      : REMark name field
DIM address1$(9,20) : REMark address line 1
DIM address2$(9,20) : REMark address line 2
DIM address3$(9,20) : REMark address line 3
DIM address4$(9,20) : REMark address line 4
DIM postcode$(9,20) : REMark postcode field
```

So, this database contains 6 fields in each record, the first being the person's name, the next four contain the address and the final field contains the postal code.
This example uses fields which have the same width (20 characters). Where this is true, it may have been easier to use one array with an extra dimension:

```
DIM name$(9,5,20)
```

In this way, the '9' represents the number of records the file can handle (0 to 9) and the '5' represents that the database will have six fields (0 to 5).
If it helps to remember which term is which, think of an address book. Each page is a 'record'. The number of pages in the book is how many 'records' the book can store. Each part of the page, each line if you like, is a field within that record.
In practice, fields in a file like this are rarely the same size. A postal code is rarely more than 8 to 10 characters, while 20 characters is really a bit short for names and addresses.
Another way of storing the data in an array is to dimension an array wide enough to hold the names and addresses in a single line, rather like columns of text. We need to work out how many characters we need in total. In this case, we dimensioned six arrays, all up to 20 characters wide, so we need 6*20 = 120 characters in total, and we can store the whole lot in an array called entry$:

```
DIM array$(9,120)
```

For this to work we need to make sure that each record (each array entry) is a fixed size and padded out to 120 characters. The best way of doing this is to pad it out with spaces:

```
FOR a = 0 TO 9 : entry$(a) = FILL$(' ',120)
```

Important: when dimensioning arrays, you should make sure it's an even last dimension, otherwise the QL may round it up for you and cause unpredictable side effects. So DIM entry$(9,10) is OK. DIM entry$(9,11) is not, and you may never be sure if the QL made each string 10 or 12 characters long.
Having all fields in a single array dimension works by virtue of the fact that SuperBASIC lets you assign data to a string slice as long as you are careful to assign the right length and pad the fields out with spaces to make them the right length and make sure that old data is deleted before you assign any updated data.
Since each field happens to be the same length in our database, we need to make sure that when we use a LET statement to put text into the array, we start the assignment at the right place and make sure we put the right length of data into the array. So to put the name field into entry$(0), the first field in the first record, we could input the name and use a LET statement to put it into the right place:

```
INPUT name$
IF LEN(name$) > 20 THEN name$ = name$(1 TO 20) : REMark too long?
IF LEN(name$) < 20 THEN name$ = name$&FILL$(' ',20-LEN(name$))
LET entry$(0,1 TO 20) = name$
```

It does have the disadvantage that the name is padded with spaces, so you'll always get back the padded 20 character length of string and you don't really know the real width of the name unless you strip off spaces at the right of the name when you read it back later. It can be a positive advantage in some cases, e.g. where data is displayed in columns and rows, like a spreadsheet or text file with text in columns. Ultimately, it is up to you which type of array method you use to store your database, depending on what is most appropriate to what type of database you build, and how the data is to be displayed.

Internal Data Formats

So far, we have used the PRINT and INPUT commands to send data to and recover data from a file. In many cases, this is fine. It writes the data to the file pretty much like text, followed by a linefeed character. It becomes a little restrictive in that numbers are actually stored as text.
Toolkit 2 introduced new commands to SuperBASIC which allow us to write and read data in what is called Internal Format. This is how the QL stores data in its own memory and can be more convenient (but harder to view, read and understand by humans!) to use when it comes to random access files. These new commands and functions (PUT, GET, BPUT, BGET) allow us to write and read data using the same format as the QL uses internally:

Strings are stored using a 2 byte value for the length, followed by the text of the string. This allows strings, in theory (memory permitting), up to about 32 kilobytes long.

Integers are stored as a 2-byte value, storing values from -32768 to +32767

Floating point values are stored as 6 byte values.

Byte values can also be written and read. Byte values can hold values from 0 to 255.

A very real advantage of this is the predictability of length of data. Using PRINT, the value 32000 needs 5 bytes plus a linefeed, whereas 99 requires only two bytes plus the linefeed. If we use internal data types, an integer value like those two always need the same number of bytes (2), so the length stays the same in a file. Likewise, floating point numbers always need 6 bytes - the length doesn't depend on the number of digits. And with text, the length is always the number of characters in the string plus the two bytes used for the length. So a string would look like this:

‹2 bytes for length of string› Text follows the length bytes

We use the commands PUT and BPUT to write such data to a file. PUT knows from the name of the variable which type it is to use. Names which end with $ are treated as strings, names ending with % are integer values and names ending with neither of those symbols are handled as floating point numbers. BPUT always sends the data as a byte value even if you use an integer% name or a floating point variable.

The command is followed by a channel number indicating which file data is sent to or read from, then the name of the variable being handled:

```
100 OPEN_NEW #3,flp1_testfile
110 LET a$ = "QL Today" : REMark a string
120 LET year_no% = 2007 : REMark an integer number
130 LET float_num = 1.5 : REMark a floating point number
140 LET byte% = 255     : REMark a byte value
150 :
160 REMark send these variables to the file
170 REMark in internal format
180 PUT #3,a$         : REMark send a string
190 PUT #3,year_no%   : REMark send an integer
200 PUT #3,float_num  : REMark send a floating point number
210 BPUT #3,byte%     : REMark send one byte value
220 CLOSE #3
```

This simple example sets up a few variables and opens a new file, then shows how to use PUT and BPUT in lines 180 to 210 to write the values of the variables out to file in QL internal format.

So what we get is as follows:
‹2 length bytes›"QL Today" ‹2 byte integer› ‹6 byes float value›‹byte value›

Note that there are no linefeed (newline) characters between them all. Try copying this file to the screen to view it and it will look pretty meaningless, with a little text you can read. This shows a little disadvantage with this method of storing data – it is not easy for humans to read, although it is great for a QL to handle!

Here is how we read the data back into the QL later:

```
100 OPEN_IN #3,flp1_testfile
110 GET #3,a$         : PRINT a$
120 GET #3,year_no%   : PRINT year_no%
130 GET #3,float_num  : PRINT float_num
140 BGET #3,byte%     : PRINT byte%
150 CLOSE #3
```

The program fetches the values of the variables from the file and prints them to the screen to prove that they are the same ones we wrote to the file previously.

Although I happened to use the same variable names, you don't have to do so, as long as you use the right type of variable. The first one must be string. The second one must be an integer, the third one must be a floating point number, and the fourth one must be either an integer (best) or a floating point variable, either can accept the one byte value. Sometimes, QL type coercion can convert from one type to another, but you are best advised to use the same type of variable where possible to avoid any mix-ups.

File Position

We can see where in the file the QL's file pointer is looking at by using the FPOS function from Toolkit 2. This returns a number which tells us how far from the beginning of the file we are. In other words, it gives us a value indicating where the next value will be taken from. Give it a channel number and it will tell you this pointer's value. As the QL can have more than one file channel open at the same time, you need to be careful you are using the correct channel number. Assuming that the file is already open:

```
PRINT FPOS(#3)
```

or

```
LET file_postn = FPOS(#3)
```

We can use FPOS in our last example to help us understand how the file is stored, by printing its values as we read each item back from the file. This can help us debug a program when we make a mistake and things don't quite work as expected:

```
100 OPEN_IN #3,flp1_testfile
110 fp = FPOS(#3) : GET #3,a$         : PRINT fp;':';a$
120 fp = FPOS(#3) : GET #3,year_no%   : PRINT fp;':';year_no%
130 fp = FPOS(#3) : GET #3,float_num  : PRINT fp;':';float_num
140 fp = FPOS(#3) : BGET #3,byte%     : PRINT fp;':';byte%
150 CLOSE #3
```

You should get a list like this:

```
0:QL Today
10:2007
12:1.5
18:255
```

Which shows that when a file is first opened, the file pointer is at position 0, zero bytes in from the beginning of the file. Then, we read in the string, which we expect to be 8 characters long plus the two length bytes. So, after we read in the value of a$, the file pointer moves 10 bytes further into the file, pointing to the next item, which is the integer value 2007. We read in two bytes for the integer

variable year_no%, which then leaves the file pointer at 12, ready to read in the value for the floating point number float_num, which in turn leaves the file pointer moved 6 bytes further on pointing at the byte value ready to be fetched to the variable byte%.
Hopefully, after that example we can now start to appreciate how convenient to use these internal storage formats are for us to use in files, because the numbers of each type are always stored using exactly the same amount of bytes in a file, and strings are always their own length plus 2 bytes for the length. When it comes to handling individual records in a large file, these "known length" elements will come in very useful.
We know how to check the file pointer position, but we can set it too, using a special option of the GET and BGET commands. If we use a backslash symbol \ after the channel number, this tells the command that the number or expression which follows is a file pointer position value. So we could change the value of the integer year_no% above and use this to alter the value stored in the file, without altering anything else. This will help explain the term RANDOM ACCESS FILE because we are able to point to any item in the file pretty well randomly and read or write that item as long as we are careful to use the correct data type. To open a file to allow us to read and write to it, we use the OPEN command:

```
100 OPEN #3,ram1_testfile
110 year_no% = 2008
120 BGET #3\10
130 PUT #3,year_no%
140 CLOSE #3
```

A note here about the various types of OPEN commands.

OPEN_NEW opens a new file. If a file of that name already exists, it will ask if you wish to overwrite it or not, as long as you have Toolkit 2 (if not, it just gives an 'Already Exists' error report). Only one channel can be opened to this file at a time to write to it – two separate programs must not try to change the data at the same time or absolute chaos could result!

OPEN_IN opens a file for input only. It is not possible to change any of the file content – attempts to do so would result in a 'Read Only' or 'Write Protected' error message. But since it is not being written to, several programs can open_in channels to input data from the same file.

OPEN opens a file in such a way that you can write to the file and input from it at the same time. OPEN_NEW creates a new, blank file, whereas **OPEN** just opens a channel to both read and write with that file. That's really the only difference between **OPEN_IN** and **OPEN** – both open an existing file, but **OPEN_IN** doesn't let you alter the file at all. If one program has a channel opened to the file with **OPEN**, no other program can do so, but other programs may be able to **OPEN_IN** a channel to that file to input something if really necessary, though this may not be a good idea as it could possibly lead to confusion. As long as programs are careful to watch out for new items added, and not to make assumptions about file lengths and so on, this may not necessarily be a bad thing.

If you have Toolkit 2, you will also have an **OPEN_OVER** command. This one deletes any existing file and then opens the file like an **OPEN_NEW** command. Use with care, of course, it is only too easy to delete a file by mistake!

Think of an example where we have a small call centre handling calls from customers. All need to be able to read information from the database. Only the supervisor can make changes to it. So the supervisor's QL opens a channel to the database using an OPEN command, while the other operators have their QLs open a channel to the file using OPEN_IN. This allows them to search for the customer's details, but to add changes of address for example, they have to put the caller through to their supervisor who is the only person to have permission (write access) to change anything in the database.

I hope that by now we are starting to get a picture of how the two types of files work. As the names imply:

* **SERIAL files** - we start at the beginning and work our way through the file in order.

* **RANDOM ACCESS files** - we can also jump back and forth by moving the file pointer about within the file, meaning we can change a specific entry in the file almost at will, and add more data to the end of the file without first having to read it all in.

Setting up a Database

A good deal of forethought is needed when designing a database using random access files, since changing the structure of a database (adding or removing fields or changing data types) may prove difficult.
We would need to assess how many fields are needed, and how large each needs to be. We also need to know which fields are to hold text, which ones integer numbers, which ones floating point numbers and so on, so that we can make a note of how much space each field and each record will take in the database, to enable us to calculate how far we need to jump from part to part.
A simple little subscriptions database will illustrate this. We decide that we need text fields for subscriber's name, four lines of address, one line for postal code, and a final number field for how many issues of the magazine is left until the subscription expires.

```
max% = 100 : REMark maximum number of subscribers allowed
DIM name$(max%-1,20)
DIM addr1$(max%-1,30)
DIM addr2$(max%-1,30)
DIM addr3$(max%-1,30)
DIM addr4$(max%-1,30)
DIM postcode$(max%-1,10)
DIM issues%(max%-1)
```

Looking at this we can work out how much space each record is to take, by assuming that each string needs its maximum number of characters plus two bytes for the string length. Integer numbers need two bytes each:

name$ needs 2+20=22 bytes per entry
addr1$ needs 2+30=32 bytes per entry
addr2$ needs 2+30=32 bytes per entry
addr3$ needs 2+30=32 bytes per entry
addr4$ needs 2+30=32 bytes per entry
postcode$ needs 2+19=12 bytes per entry
issues% needs 2 bytes per entry

When we use PUT to send these field variables to the file, we will need a total of 22+32+32+32+32+12+2 bytes (164 bytes in total) per record. In other words, every full entry in the database will need 164 bytes. As we set a maximum of 100 entries (the variable max%), a completely full database will need 164*100 bytes, or 16,400 bytes. This is the beauty of using PUT and BPUT commands, everything can be handled as nice neat predictable lengths of data - so we can jump around and fetch any data we like using multiples of the field length. To go to the second entry in the database, we simply skip the first 164 bytes. In other words, we skip 152 bytes times the record number less one. To get to the 10th entry:

bytes_per_record * (entry_number-1)
164*(10-1) = 164*9 = 1476

So, assuming everyone's membership number is actually their position in the database (the first is 1), we can write a line or two of SuperBASIC to calculate where their entry in the database should be:

```
INPUT"Membership Number › ';memb_num
postn = 1476 * (memb_num-1)
OPEN_IN #3,flp1_database
BGET #3\postn
```

and we are quickly at the member's records, which we can read in with a few GET commands, into suitable variables.

```
GET #3,n$ : REMark name
GET #3\postn+22 : REMark next field starts here
GET #3,a1$ : REMark address line 1
GET #3\postn+22+32 : REMark next field starts here
GET #3,a2$ : REMark address line 2
GET #3\postn+22+32+32
GET #3,a3$ : REMark address line 3
GET #3\postn+22+32+32+32
GET #3,a4$ : REMark address line 4
GET #3\postn+22+32+32+32+32
GET #3,p$ : REMark postal code
GET #3\postn+22+32+32+32+32+12
GET #3,i% : REMark issues left
```

By now, I hope you can see that accessing any part of the data is as easy as calculating its position in the file and fetching it from there. It is important that even if the fields are shorter than their maximum possible length that we pad out the file with enough spaces or zero codes to make sure the record is always the right length even if this seems a little wasteful.

When adding a new record to the file we point to the end of the existing data, remember that position, send the same number of bytes as the maximum record size, point back to where this new record started, and then PUT the data in the right places as required.
An easy way to set the file pointer to the end of the file is to use FLEN(#channel_number):

```
fl = FLEN(#3)
BGET #3\fl
```

We can then add enough bytes of space by printing the right number of spaces or nulls to the file. In this case, our records were 1476 bytes long:

```
PRINT #fl,FILL$(" ",1476);
```

Then, point to the new position again to start PUTting the new record there:

```
BGET #3\fl
PUT #3,new_name$
BGET #3+22,new_addr1$
PUT #3,new_addr1$
```

and so on.

Having a database in a known layout like this makes it fairly easy to search through it as well. Using our subscriber database example above, suppose that the subscriber called us, but didn't know his membership number. We could then search for his postcode –

```
INPUT"Search for which postcode?";pcd$
FOR a = 0 TO number_of_records-1
  REMark work out where each record starts
  fp = a*1476
  REMark now work out how far into the record is the postcode
  BGET #3\fp+22+(4*32)
  GET #3,postcode$
  IF pcd$ == postcode$ THEN
    REMark we have found the subscriber details
    REMark extract them here
    ...
    EXIT a
  END IF
END FOR a
```

Comparisons and Conclusions

Learning about how random access files work is good background knowledge for using Archive, since it works in a similar, though somewhat more advanced way. Many database programs you can find for the QL such as K-Base and EasyBase also use these types of principles and so it's all good background knowledge, and would also stand you in good stead when it came to writing customised databases using the DataDesign or QBase database programming languages. Although not all database programs use fixed size records for random access files like this (Flashback and DataDesign are notable exceptions which use their own unique format flexible record lengths and so on), the grounding in knowing what fields and records are, and a little experience of writing your own code will really help you understand what other free and commercial database programs do and to understand the possibilities of what all these programs can really do if we can achieve fairly sophisticated things just using a few lines of SuperBASIC.

Bits, Bytes and Nybbles

by David Denham

Don't you just find some of these computer terms hard to master?

Although I'm a bit more confident with them now than I was a few years ago, one of the hardest things for me in the beginning was terms like bits, bytes, nybbles, words, long words and octets. Then came nasties like hexadecimal, binary and so on.

If I once had problems with these terms, I'm sure others must have too, so let's see what I can do to put pen to paper to try to explain all these.

Computer memory, like all things digital, is based on ones and noughts, with all values in a computer being represented by groups of ones and noughts. Any single digit can only have one of the two possible values of 0 or 1, so they are referred to as "binary" values. The binary counting system doesn't use anything other than the digits 0 or 1 (no 2, no 3 and so on), but this is a bit useless by itself, so computers use groups or clusters of these binary digits for other numbers.

BIT - one single binary digit, the basic "building block"

BYTE - a group of 8 bits, holding binary values from 00000000 (0 in decimal) up to 11111111 (255 in decimal). The individual bits have equivalent decimal values, from left to right, of 128, 64, 32, 16, 8, 4, 2 and 1, which are used in combination to make up values from 0 to 255 in decimal, so for example the value of 130 would be represented as 10000010 (128+2). Don't worry if you don't understand this - binary numbers are not the easiest of subjects!

WORD - a group of two bytes, or more correctly, 16 binary digits.

LONG WORD - A group of four bytes, or more correctly, 32 binary digits.

NYBBLE - half a byte, or a group of four bits. 4 binary digits are enough to hold a value of 0 to 15 in decimal.

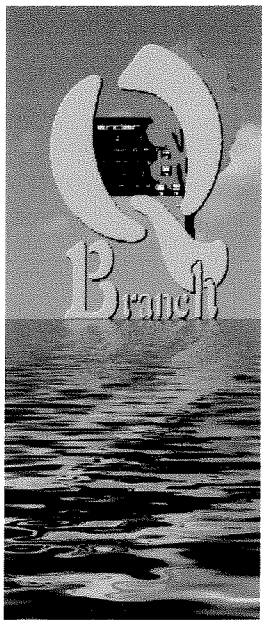
BINARY - a counting system based on powers of 2. In other words, each bit in a byte is twice the value of the preceding one when you think of the equivalent decimal number.

HEXADECIMAL - A counting system based on base 16 numbers. More information below.

OCTET - group of 8 bits, not necessarily in the same byte (e.g. may be half of one byte plus half of the next byte). You tend to come across this term a lot when discussing the technical side of the internet and data communications.

All computer based counting systems tend to be based on powers of two - 1, 2, 4, 8, 16, 32, 64, 128 etc.

Decimal is well suited to human counting - last time I counted them, I had 10 fingers and 10 toes. But computers tend to prefer multiples of 2, and 16 is such a number, so it makes hexadecimal (a numbering system based on the number 16) a good system for working with computers, since hexadecimal numbers can be stored in a nybble (half a byte). Values from 0 to 15 can be stored in four binary digits, and multiples of four bits can be held conveniently in bytes, words and long words. This is why hexadecimal numbers (or "hex" for short) is so widely used when program-



Feeling out on a Limb?
Reach out for QBranch
Suppliers of Quality QDOS/SMSQ products
Hardware and Software

20 Locks Hill
Portslade
Sussex
BN41 2LB
UK
Tel: +44(0)1273 386030
Fax: +44(0)1273 430501
Mobile: +44(0)7836 745501
email : sales@qbranch.demon.co.uk
www.qbranch.demon.co.uk

We can accept payment by Visa, Mastercard, Maestro and Switch. We also accept sterling cheques drawn on a UK bank. or payment direct to our bank:
Natwest
Sort code 60-17-01
Acc. 84534826
BIC : NWBKGB 2L
IBAN :
GB34 NWBK 6017 0184 5348 26

QDT

The Ultimate QL Desktop
v1.02
£41.50 + p&p
£2.00 UK / 2.50 EU / £ 3.00 RoW
Needs SMSQ/E & extended colours to run

EasyPTR v4

Pts 1&2
Major Rewrite!
£42.00
Produce High Colour pointer-driven menus from SuperBASIC
Also on CD Rom

Text 87 £ 79.00

High Colour Patch (all systems) £9.00
ESC/P2 drivers £ 26.00 / Fountext £39.00
Typset94 / 2488 drivers £29.00 ea.

SMSQ/E and QPC 2

SMSQ/E for Gold Card / Atari / QXL / Q40
Aurora High Colour Driver included with Gold Card Version
Now Only **£15.00 + postage**

QPC 2 v 3.33

Now only £ 42.00

Upgrade Prices:
from v 3.xx Free
from v 2.xx £13.90
from v 1.xx £34.00

Delivered on CD ROM with Manual as Adobe Acrobat File

QPCPrint

Print to any Windows Installed Printer from Your QL Programs !
£27.00

Programs upgraded to high colour version

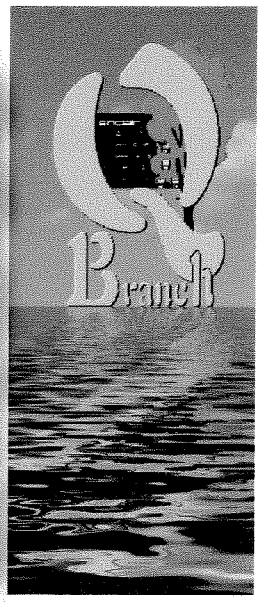
QMake / WinEd / QPAC 1 / QPAC 2 / QSUP / SuQcess / Disk Mate 5
Send old master disk and £1.60 per disk to cover costs

Free !

Paid Upgrades	
QD 98 to QD 2003	£10.50
QSpread 2001 to QSpread 2003	£10.50

Return Master Disk with order

QBranch have now combined their fax and phone numbers. New phone number is +44 (0) 1273 430501
At the time of going to press we have some second user hardware including a Super Gold Card - email for details.



Programs

Utilities

Fifi2	File Finder	£21.00
QSup	Utilies	£30.00
QSpread	Spreadsheet	£51.00
Cueshell 2	File Manager	£15.00
QPAC 2	File Manager & Utilities Package	£42.00
QPAC 1	Calendar, Clock, Calculator, Sysmon	£22.00
QLoad/Qref	Fast load for QDOS	£15.00
QTYP2	Spell Checker	£31.00
QLQ	Printer Utility	£30.00
SuQcess	Database	£28.00
Q-Route	Route Finder	£25.00
Knight Safe3	Backup Program	£35.00
Fractal Collection	Fractals	£35.00
QCount	Accounting	£25.00

Programming

QD 2003	Text Editor & More	£ 49.00
QBASIC	QLiberater to QD Link	£ 15.00
QLiberator	Basic Compiler	£ 50.00
QD + QBASIC		£ 63.00
QD + QBASIC + QLiberator		£104.00
QPTR	Pointer Toolkit	£ 32.00
MasterSpy	Fast Text Editor	£ 30.00
QMake	Assembler Tools	£ 18.00
QMon/Jmon	Monitor - Upgrade only	£ 22.00
BASIC Linker	Basic Library Linker	£ 22.00
Disa 3	Dissassembler	£ 34.00
QMenu	Menu Extensions & tutorial	£ 16.00
Easyptr v4	Toolkits & Programming Extns	£ 41.50
Easyptr v4	Part 3 C extensions	£ 14.00

HARDWARE

We have a rotating stock of both new and second user hardware. It is best to call or email us for details of what is available.

Recycled Items
(when available)

Super Gold Card	£email
Gold Card	£email
Aurora	£ 60.00
QXL	£ 35.00
superHermes	£ 65.00
DiRen Keyboard Interface	£ 15.00
Qplane	£ 5.00

New Items

Aurora	£70.00
Aurora w/8301 & Minerva	£80.00
8mb Rom Disq	£98.00
4mb Rom Disq	£65.00
2mb RomDisq	£39.00
mPlane	£34.00
MCplate	£ 6.50
Various braQuets	£ 8.00
Gold /Super Gold Card Batteries	£10,00

We also have a collection of standard QLs, QL Power supplies and some QL books.

Cables for the Aurora, Qubide and Super Gold Card ROMs and other QL accessories are also available from us.

Call for details

ming computers in assembler or machine code despite it being relatively hard for humans used to normal decimal numbers to understand!
A hexadecimal number can have values from 0 to 15, with letters representing numbers higher than 9:

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

As with decimal numbering, when we need a higher number than one digit can hold, we add a new digit to the left. So while 15 is the highest decimal number which can be represented by a 1 digit hexadecimal number, 16 becomes 10 which is pronounced as "hex one zero" rather than "hex ten" or "ten" to avoid confusion with decimal numbers.
Bytes can hold decimal values from 0 to 255 inclusive, which can be represented by two hex digits from 00 to FF. It is accepted practice to indicate that a number is a hex value by preceding it with a '$' symbol, e.g. $00 or $FF. Likewise, a binary value is usually preceded with a '%' symbol, e.g. %00000000 or %11111111

Toolkit 2 has a couple of very useful functions to convert between decimal and hex values, and between decimal and binary values - these are HEX, HEX$, BIN and BIN$

```
PRINT HEX('FF')
```
converts hex FF to decimal, which gives 255.

```
PRINT HEX$(255,16)
```
converts decimal 255 to hexadecimal, using 16 bits or four hex digits, which gives 00FF

```
PRINT BIN('1100')
```
converts a binary number to decimal, which gives 12.

```
PRINT BIN$(15,8)
```
converts decimal 15 to binary, using 8 binary digits. This gives 00001111

If you wish to convert between binary and hexadecimal, the easiest way is to convert the binary value to decimal first, using BIN, then use HEX$ to convert the decimal value to hexadecimal. Likewise, to convert from hexadecimal to binary, convert to decimal first using the HEX function, then use BIN$ to convert the decimal value to binary.

Binary and hexadecimal are difficult concepts to grasp at first, but you'll need a basic grasp of them before you try to tackle assembler or machine code programming.

Addresses

The QL's memory is based on groups of 8 bits, known as bytes, and these are organised in a long chain known as "addresses", starting from address 0 and each byte after that being one location higher in memory.
As we are talking in computer terms here, the amount of memory on a QL is measured not in units of 1,000 bytes as you might expect, but units of 1,024 bytes, called kilobytes, because 1,024 is a power of 2. And if the computer had millions of bytes, this would be units of 1,024 kilobytes - known as megabytes.
The QL has 48 kilobytes of ROM space (ROM stands for Read Only Memory, because the program it contains is fixed and cannot be altered - once Sinclair programmed those ROMs, you cannot reprogram them, all you can do is change the ROM completely). 48 kilobytes is 48*1024 bytes, or 49,152 bytes.
The QL also has 128 kilobytes of RAM (Random Access Memory - the non-permanent type, it loses the content every time you switch off or reset the QL).

Memory Storage

Let us have a quick look at how the QL stores information in memory. We've previously seen that a single byte can hold values from 0 to 255 (decimal values). We'll look at how it stores the word "Hello" for example.
The letter H is stored at the lowest address. For the sake of argument, we'll assume this starts at address 200,000 although in reality it could be just about anywhere in memory, depending on what else the QL is doing, what size of program

it is running and so on. Each letter is stored in consecutive bytes of memory. Picture each memory address as being a little matchbox which can hold numbers from 0 to 255. The QL doesn't store the letters as such, it stores the CODE value of each letter. These are the same codes that the functions CODE and CHR$ understand in SuperBASIC. For each letter, the code is found as follows:

```
PRINT CODE("H")  gives 72
PRINT CODE("e")  gives 101
PRINT CODE("l")  gives 108
PRINT CODE("o")  gives 111
```

So we need five matchboxes starting at address 200,000 to store the text "Hello":

Address	Decimal value	Letter
200,000	72	"H"
200,001	101	"e"
200,002	108	"l"
200,003	108	"l"
200,004	111	"o"

This is how text is stored on the simplest level. Numbers can get a bit more complicated and we'll pass on that for now!
The basic idea is that the first letter is stored at the lowest address, the second letter at the next highest address and so on.

PEEK and POKE

These commands work on the actual values held at a given address in the QL memory.

```
PRINT PEEK(location)
```

returns the value of the byte at a given address. If you are using an original QL, the picture you see on the screen is stored in memory starting at address 131,072 which is the top left corner of the screen. A QL screen needs 32,768 bytes of the computer's memory. You can use PEEK to read values from the screen memory, although it is not very good practice.

```
PRINT PEEK(131072)
```

tells you what value is stored at the top left of the screen. Likewise

```
POKE 131072,a_value
```

will change it. It is even worse practice to use POKE to place values directly into the screen. At best, the colour of part of the screen may change, at worse you could crash the QL especially if you make a mistake.
While PEEK and POKE work on one byte values, there are versions of these commands to work on word values and long word values. These are called PEEK_W, POKE_W, PEEK_L, and POKE_L. Since the QL has 8 bit memory, and these commands work with 2 byte or 4 byte values, what they do is to work with two or four consecutive addresses.

Conclusion

Do not be too disheartened if the last part of this article was a bit heavy going. I hope you will have found something of value in it and that it's made some of those computer jargon words a bit more meaningful to you.

The Mercator Map Projection

by Hugh Rooms

Introduction

This is written from the point of view of someone living in the British Isles, i.e in the region 50° to 60° North latitude, and 2° E to 12° W longitude. For those who live elsewhere, I apologise for this insularity; I hope that it still makes sense and that you can easily translate it all to your own coordinates.
While I was working on the GPS system described in previous articles, I found a lot of material on the Internet about the problems of relating the coordinates obtained from the satellites to positions on the Earth, and then on to maps, especially those produced by the Ordnance Survey (OS) in Britain (1) (numbers in brackets refer to the references at the end of the article). I found that I really needed to go back to basics and start with an understanding of the method used by the OS to represent the curved surface of the Earth on a flat sheet of paper, described as a Transverse Mercator Projection (TM). This article deals with the Mercator Projection (MP) found in world atlases; and TM, which is an extension of it, will follow later.
The calculations involved in the official literature seemed horrendous, so I had to find an easier approach to start with. It seems that what causes most of the trouble is the use of an ellipsoid model for the Earth, so, to make it simpler, I decided to try a spherical model instead; indeed I think that it would not be possible to follow the more exact calculations without understanding this approach first. Thus my ambition became to

understand the MP and TM projections applied to a spherical globe. I easily dug out the equations for this simpler case, but I could not find their derivations anywhere, so I had to work them out for myself: the maths I eventually came up with is given in the two appendices. Here I must gratefully acknowledge the strenuous and extended efforts Geoff Wicks made to accommodate the maths into this article, I could not have produced it without this help.

Introduction

When I first started I had only a vague idea of exactly what a map means to the person drawing it. As map users we are mainly concerned with the symbols showing what is there, and the directions and distances that relate them. To the cartographer, as far as it concerns the maps I deal with here, it is just a mass of *(x,y)* points on the paper, each corresponding to a *(longitude,latitude)* pair for something on the Earth's surface; the symbols and lines joining them are added later. This difference is even more apparent in that longitudes and latitudes are actually angles, but they are plotted on the map as distances. This caused me some confusion, until I realised that angles and Great Circle distances (which I will explain later) on the (spherical) Earth's surface are equivalent, a kilometre is one minute of arc in any direction – 1/60th of a degree – to a close enough approximation for us here. Because of the distortion this does not apply to a map, where everything is measured as, say, millimetres on the paper. The intention is to draw the map in such a way that the user gets accurate enough distances and bearings when he transfers his measurements from the map back to the real world.

Cartographers, over centuries of mapmaking, have devised many ways of projecting the curved surface of the Earth on to the flat sheet of a map. All necessarily suffer from some distortion, but one of the most useful over small areas is Mercator's Projection (MP), devised in the 16th century in the Low Countries, and popularised by Gerard Mercator (1512-1594) (2). It has a bad reputation because, when used for a World map, figure 2, it gives extreme distortion of size towards the Polar regions. However not many people using OS maps (and those of other national surveying bodies) will realise that they are drawn using a modified form of MP, called a Transverse Mercator Projection (TM) (3).

My ultimate aim in this project is to have QL programs that will convert between GPS coordinates and OS National Grid references. The professionals want to get the greatest accuracy possible in the conversion, perhaps less than a metre; I would be happy with an accuracy of about 100 metres, which I think may be attainable without going to the ellipsoid model, but I still haven't reached a conclusion on this. Any success or otherwise will be reported in a third article.

So this article describes how a set of points on the Earth's surface, defined by their latitudes and longitudes, are translated into *x* and *y* coordinates plotted to make a map, using MP. The second article will show how this is converted to a TM projection, as used by the OS, and also for a standardised world wide grid of maps in the Universal Transverse Mercator projection(UTM) (5). Each article has a QL SB program to go with it but bear in mind that I am using a spherical model for the Earth, so the results for TM will be more approximate than the model the OS uses. Accurate formulae and conversions for PCs are available on the OS web site (1).

Mercator's Projection

The Mercator projection is often described as being obtained by wrapping a sheet of paper round the Equator of a suitably scaled transparent globe, marking off the meridians as vertical lines where the paper touches them, and projecting the parallels of latitude as if by a light at the centre of the globe so that they are spaced further and further apart on the paper towards the poles. Although this is an attractive model, and

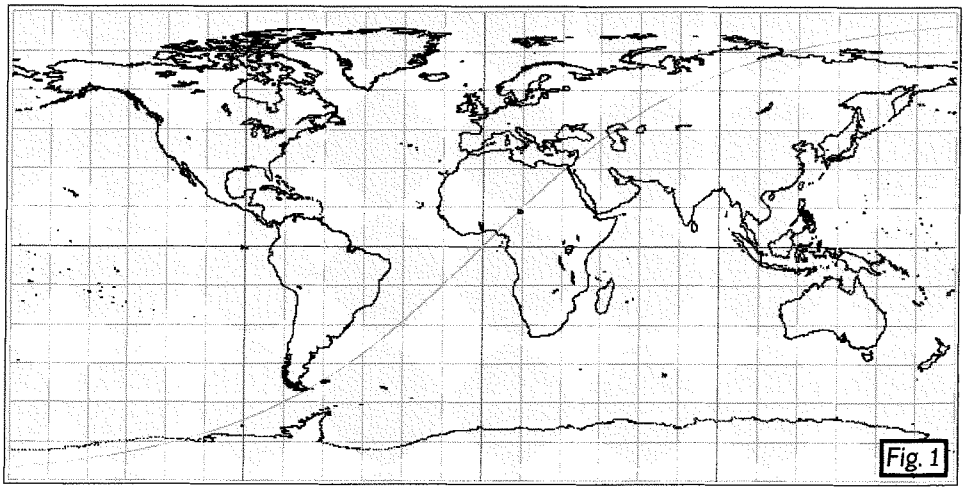


Fig. 1

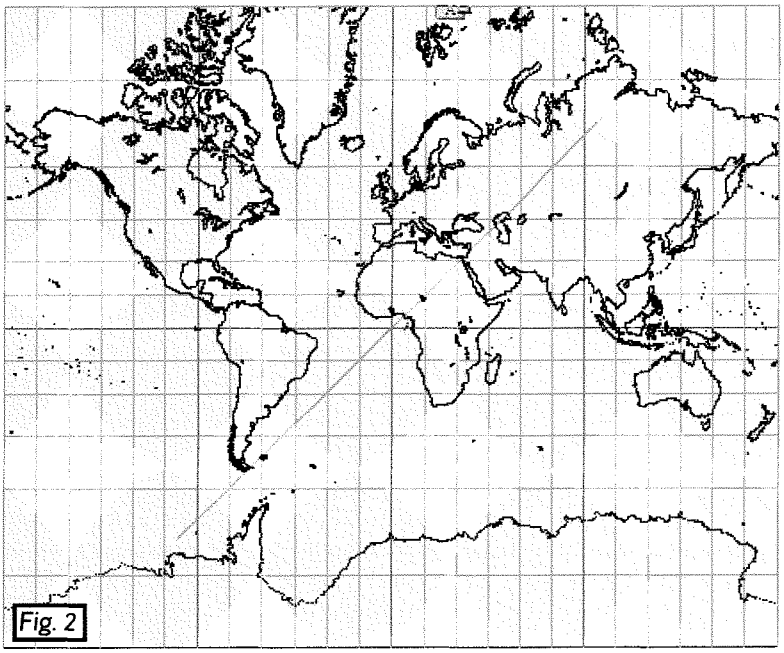


Fig. 2

gives a qualitative picture of MP it produces a different spacing for the latitudes; the MP spacing is much more subtle.

The basic idea of MP is to preserve bearings: using MP all bearings are correct, however big the area mapped, so it is useful for navigation charts, and for gunnery (i.e. ordnance); and this also conveniently means that, over a small area (and this "small area" can be as big as the U.K), *proportions* i.e. *shapes*, are quite accurately preserved. Lines of constant longitude, i.e. meridians, are drawn on the map as parallel vertical lines. Lines of constant latitude are parallel on the Earth, and remain so on the map as horizontal straight lines perpendicular to the meridians.

The real meridians get closer together as they converge towards the poles, but, because on the MP map they remain the same distance apart, distances measure E-W on the Earth become longer and longer on the map as their latitude increases; so, to keep the proportions correct over an area, N-S distances, the spacing between latitudes, must be extended in the same ratio. That long sentence is the essence of the Mercator projection, what remains is to calculate how to do it.

The mathematics involved in the spherical model is mostly fairly straightforward trigonometry, with a little bit of calculus for a bonus. I have tried in the Appendices to explain it all in a way that I hope will be useful to those whose trigonometry is as rusty as mine was when I started out on this. My first reaction when seeing a closely packed page of maths is that I can't be bothered to unravel it to find out what it means; usually there are missing leaps of "obvious" (to the writer) working that I have to struggle to fill in again, and it all gets very tedious. I have tried in those Appendices to list all the steps of the processes, but I have quoted some of the standard formulae I use instead of proving them all, that would make it far too long.

I hope you can take a few minutes to look at it before you dismiss it as too esoteric; I intend that even if school maths is a distant memory, that what is here will 'click'. I was once an engineer – a long time ago – and practiced what I call "pragmatic mathematics" or "pragmaths", that is: use it, don't worry too much about the ifs and buts unless they aid in understanding; get a result, if it looks OK then try it out. That's how I have approached it here.

At the top left of figure 5, which goes with Appendix 1, is a sideways view of a meridian, as if we had taken a cross section through the Earth, and we see that the radius of the parallel of latitude at φ degrees North is $R*cos(\varphi)$. I use the programmer's asterisk * for multiplication, instead of times or a dot (x or.) which could be confusing in text, although I keep the dot as a product in some of the diagrams.

Before considering the rest of Appendix 1, let me introduce my idea of a *Whatever* as a unit of measurement. As it suggests you can make a Whatever whatever you want, the only rule is that you must be consistent. By making the radius of the globe "one Whatever", we can then drop it from the calculations until we get to the point of working out the scale of the map: engineers will recognise this as using "per unit" calculations.

The x distances on the map are direct plots of the angles represented by longitudes: the scale of degrees E-W stays the same at all latitudes. But the convergence of the meridians means the distance between them measured in kilometres *decreases*, since the radius of the parallel at a high latitude φ is just $cos(\varphi)$ times *Whatever*, reduced from one *Whatever* at the Equator. True E-W distances on the Earth, when they are represented on the map, have to be *increased* in the same proportion, that is $1/cos(\varphi)$, or $sec(\varphi)$. To preserve shapes and bearings, North-South distances must be increased by the same amount. However, as explained in the Appendix, we cannot simply plot the latitude φ as $sec(\varphi)$; we have to take account of all the other latitudes already increased as we go from the Equator to our chosen point.

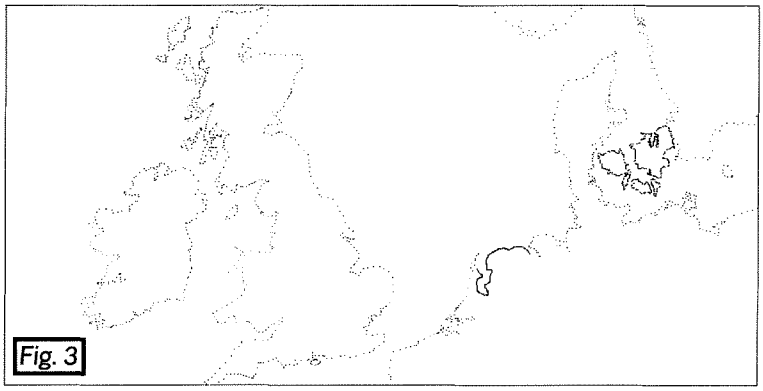


Fig. 3

To go further we need a little bit of calculus, so the rest of Appendix 1 is devoted to that, and states the resulting formula for the latitude: $y=ln(tan(\varphi/2+\pi/4)$; see the Appendix for the meaning of the other symbols. There are at least four other versions of this formula you may come across in other texts (2).

I wrote a SB program to test this all out. The results are in figures 1 to 4. After a lot of frustrated searching I found a set of data coordinates for the world's coastline on the Internet (6) and used that to draw the maps. Figure 1 is drawn the most obvious way, without any attempt to reduce distortion, on a rectangular grid using the same scale for both latitude and longitude. Also on here is a line on a constant bearing of 45 degrees. Because the meridians, which taper towards the poles, are spread out on the map, this line is curved, so you could not easily lay off a bearing on this map for navigation. The spreading meridians also account for the apparent squashing of the land masses towards the poles. Figure 2 uses the same data, but now the latitudes are spaced out using MP, resulting in a different distortion, but the constant bearing line is now straight. I have included figure 3 to show the nature of this data. Figure 4 uses a different set of data, more about that later.

For navigation the shortest route to follow, e.g. for a ship on the ocean or an aircraft, is a *Great Circle*, which is the line formed at the surface of the Earth by an imaginary section that goes through the centre, such as depicted on the left of figure 3. Meridians and the Equator are Great Circles, but all other lines of constant latitude are not. A Great Circle, which can be at any inclination to the Equator, is the shortest distance between two points: but inconveniently not the same as a line of constant bearing, called a *Rhumb* or *Rhumb Line* by navigators and a *Loxodrome* by mathematicians, which would be easy to plot, and easy to steer. Drawn on a sphere a Rhumb forms a graceful spiral converging on the poles, approaching ever closer but never actually getting there (4). You can see the difference by looking on a globe at Paris and Vancouver. For a constant bearing route, Vancouver, 49° 30' N, is pretty well due West of Paris, 48° 50' N, distance 4934 Nautical miles (Nm). To fly directly there on a Great Circle route you would head NW from Paris, pass over Greenland, and approach Vancouver from the NE, but fly only 4243 Nm, nearly 700Nm shorter.

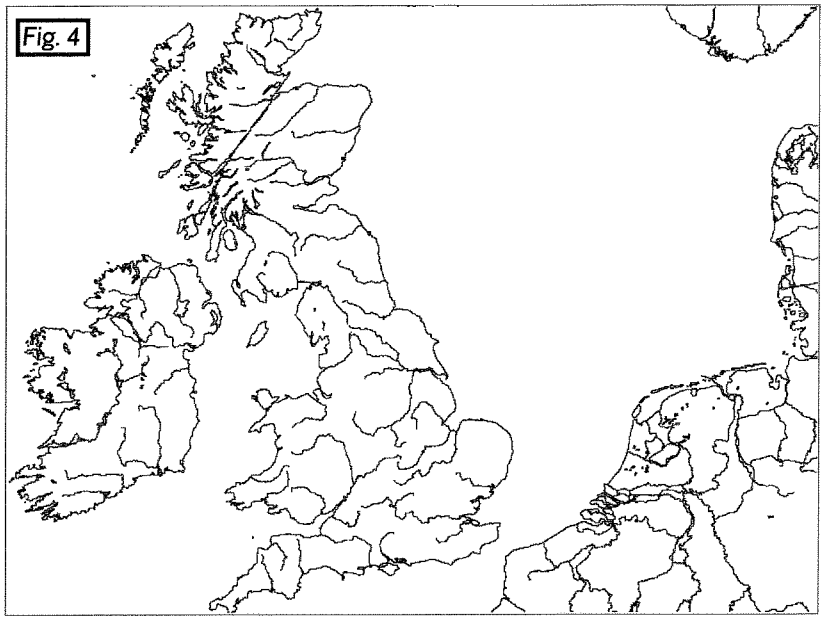
Because the heading for a Great Circle route is constantly changing, it is often more convenient to fly or sail a series of short, constant bearing, legs that approximate to it. Books on navigation give formulae for getting the bearing to set off on a Great Circle route; then, after a suitable interval the position is checked, and another leg calculated on the same Great Circle route but from this new starting point. I'm not a practical navigator, just interested in the way it works. In these days of computers and GPS, navigation must be a subtle and complex business. I have recently come across a program that analyses this for ships (8); it looks brilliant, although I haven't had time to explore it properly yet. It claims it "may be for the professional navigator what the spreadsheet was to accountants."

Comments on the Program

I hope that most of what is going on is apparent from the REMarks.

There's a choice of three maps by typing 1, 2 or 3 at the prompt: I have left out any input checks to save space. You can change the size in pixels of the Window to suit your screen: the SCALE takes care of things by being given the height in degrees, then the width is taken care of by the proportions of the window. Remaining parameters are the longitude and latitude in degrees of the bottom left of the window. You might expect that the height of a world map would be 180° pole to pole, and the bottom left at -90°, but it must be extended to allow for the distortion produce by MP. I use *asprat* to adjust *x*-values to make a square look square on my display, you may have to alter its value.

Line 350 you may need to change to suit your own data. The assumption is that it is a text file where each line has a *latitude* then a *longitude* as fixed or floating point numbers. The format must be so that line 470 and 480 read and separate them, and it is then simply a matter of plotting this point suitably scaled as in line 520.



Longitude is scaled by *asprat* and latitude by the FuNction *Merc*. One brilliant property of SB is that points plotted outside the window are simply ignored, so there is no need to check this for the GB choice, but it does take a long time to plot as most data is off the screen: it is included to show the nature of the data I used. I could check and not plot points off the screen but I would expect that to take just as long to execute.

That's all there is to the main program. For the subroutines: in *Merc* a longitude is expanded using the formula derived in Appendix 1. And for the rest: more than half the program is taken up with drawing the grid of meridians and parallels, and the line of constant bearing. The latter is a plot of opposite corners of small squares with the same increment of (suitably scaled) latitude and longitude to give a 45° heading. For the MP map I found that attempting to use the formula just meant applying it and then undoing it, so I settled on the assumption that it is a straight line .

DATA

Of course to draw the maps you need a file of data – a list of the latitudes and longitudes of enough points to make reasonable outlines, without nasty gaps. I did think of producing a modest set for myself, but that was far too long-winded and boring, so I spent some time looking for data on the Internet, eventually finding a set that I could download (6). It's not a particularly good set, an attempt to draw lines between the points showed up a lot of spurious data, but as a world map the points merge to give the appearance of solid lines. Later I searched again (and again and again – it's surprising how well this data is hidden) and came across the CIA data (7).

Unfortunately all these files are huge – the one I have used for the maps shown is 564 Kb, and defines over 80000 points, far too much to reproduce here. And the CIA data is 30Mb compressed, expanding to over 120Mb. My system, with a dial-up modem, takes five minutes per Megabyte ...

So you're on your own here, if you can download the data then you can plot the maps for yourself.

However Geoff Wicks has downloaded the CIA data and kindly sent me a copy, already converted to a format to suit my program. For comparison figure 4 uses the CIA data for Europe over a similar area as figure 3. It includes national boundaries, rivers and lakes as well as coastlines.

Final Remarks

I enjoyed all that, and the investigation that got me here. But it is just a summary of a long fumbling quest: I feel rather like someone who has solved a maze, and triumphantly presents the result as an inevitable and obvious wiggly line leading from the entrance to the exit, missing out the times spent floundering about, getting lost, and all the false trips down dead ends that were made along the way. I am pleased to have got through, I hope you find it pleasing too.

References

(1) A wonderful resource for all sorts of map related material.

(2) "Mercator: The man who mapped the planet" by Nicholas Crane ISBN 0-753-81692-X

(3) This is a source for Mercator formulae and some examples of maps drawn using them, but of course no derivations: Eric W. Weisstein. "Mercator Projection." From MathWorld – A Wolfram Web Resource.
http://mathworld.wolfram.com/MercatorProjection.html
(It asks to be cited like that)

(4) "Constant Headings and Rhumb Lines", an article that has a beautiful picture of a Loxodrome at:
http://www.mathpages.com/home/kmath502/kmath502.htm

(5) A comprehensive description of UTM.

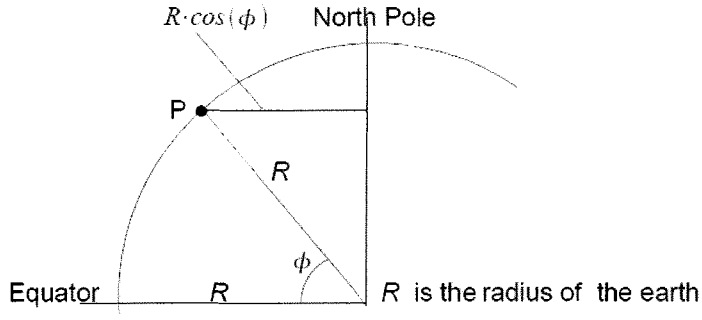
(6) **ftp://ftp.blm.gov/pub/gis/coast.zip**
564 Kb
A List of over 81,000 Coordinates I used to draw the World maps.

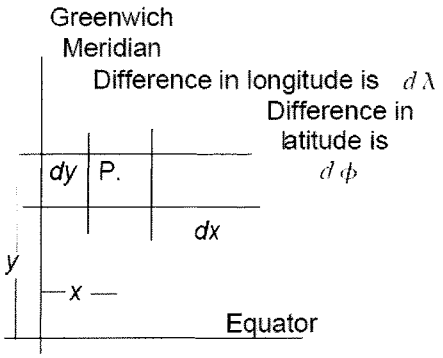
(7) World data in a 30Mb .zip file.

(8) I downloaded the 2m_nav system from this site, about 8.6Mb for Windows – a comprehensive aid for professional navigators. It will take me weeks to explore it thoroughly. I am pretty certain he uses (7) for his maps.

The Mercator Projection

The MERCATOR PROJECTION





On the left we have a cross section of the Earth along the meridian containing the point P at (λ,φ), i.e.: longitude λ (not shown), and latitude φ. We see that the radius of the parallel of latitude is reduced from R at the equator to $R*cos(\varphi)$ at P. To make the meridians on the map parallel all the way up from the equator, at this latitude we have to increase any East-West dimensions on the Earth by $1/cos(\varphi)$ i.e. $sec(\varphi)$. I assume that we apply suitable scales so from now on we can work in whatevers, and forget R and other factors in the following formulae. Angles can be in degrees or radians, as long as we are consistent, but see the caveats at the end of this note.

The right hand diagram shows the representation of the point P on a map, at (x,y) where x and y are the scaled representations of the longitude and latitude, respectively, of P , as seen looking from the left in the first figure. We use x to measure horizontally on the paper, and y vertically, the same as for drawing graphs.

The small area on the globe, greatly exaggerated in the diagram, bounded by two parallels of latitude and two meridians, at P, will become the rectangle dx,dy on the map, and its width $d(\lambda)$ on the Earth will expand on the map to a width $sec(\varphi)*d(\lambda)$ on the paper. Since $sec(\varphi)$ increases with φ on the globe, the top of the area will expand more than the bottom on the map; to avoid this we have to imagine $d(\lambda)$ and $d(\varphi)$ to be so small that this distortion does not apply to any measurable extent, and we can apply the same $sec(\varphi)$ to the whole of the area. For longitude, it may already be obvious that $x = sec(\varphi)*\lambda$ but it is useful to obtain it the way we will need later for the latitude, that is: we add up the widths of all the small areas from the Greenwich Meridian to P:

$$x = d(\lambda_0)*sec(\phi_p) + d(\lambda_1)*sec(\phi_p) + d(\lambda 2)*sec(\phi_p) + \ldots + d(\lambda_p)*sec(\phi_p)$$

i.e. the sum of all $sec(\varphi_p)*(d(\lambda_n))$ with n going from 0 at Greenwich to p at P. But $sec(\varphi_p)$ is a constant as φ_p is the latitude of P, so:

$$x = sec(\phi_p)*(d(\lambda_0) + d(\lambda_1) + d(\lambda_2) + \ldots + d(\lambda_p))$$

as we expected.

To maintain proportions, the height of the area $d(\varphi)$ must expand on the map in the same ratio, to $dy = sec(\varphi)*d(\varphi)$, and similarly for every dy from the Equator to P, so:

$$y = d(\phi_0)*sec(\phi_0) + d(\phi_1)*sec(\phi_1) + d(\phi_2)*sec(\phi_2) + \ldots + d(\phi_p)*sec(\phi_p)$$

Now the $sec(\varphi_n)$ are all different so we cannot simply make y equal to $\varphi*sec(\varphi_p)$: we must do the addition bit by bit. This summation of the dys is a classic case for calculus, or, to be precise, integration, which will nicely add up the increases for us, for all the vanishingly small areas between the Equator and P. Written as an integral it is:

$$y = \int_0^{\phi} dy = \int_0^{\phi} sec(\phi)\,d\phi$$

The "long S" means "sum" and the 0 and φ at its ends are the limits of the summation. It means "*add up all the small increments dy of y, that is* $sec(\varphi)*d(\varphi)$, from 0 to φ *when all the* $d(\varphi)$ *are made vanishingly*

small". I must admit I could not evaluate this myself without help, but the result is given in textbooks as:

$$y = ln(tan(\phi/2+\pi/4))$$

I was not happy to leave it like that, I had to verify the result, and my proof is given in Appendix 2. In this formula *ln* is *"the natural logarithm of"* or *"logarithm to the base e"*. The symbol *e* is ubiquitous in mathematics, like π, but also like π can be treated like any normal number: its value is approximately 2.71828.
We can also use calculus for the longitude:

$$x = \int_0^{\lambda} dx = \int_0^{\lambda} sec(\phi)\,d\lambda$$

giving $sec(\varphi_p)*\lambda$ as before.
A note on logarithms. If $p = a^y$ then we have $y = log_a(p)$, read as *"y is the logarithm (or log) of p to the base a"*. This is the definition of a logarithm. Correspondingly *p* is the *antilogarithm* or *antilog* of *y*. The base *a* can be any number but *e* turns up a lot in mathematics. For example 9=3*3=3**2, where ** means *"raised to the power"*, so $2=log_3(9)$. Note also that $3=log_3(27)$, $9*27=243$ and $5=log_3(243)$: it is no coincidence that in the logarithms 2+3=5. In this way multiplication can be done by adding logarithms and taking the antilog of the result. This extends to division, by subtracting logs, and to numbers with fractions; and, with base $a=10$, is the basis of a slide rule and log tables, and was the way engineering and scientific calculations were done until calculators and computers came along and took the fun out of it.
These formulae, for *x* and *y*, with suitable whatevers, can be used directly to plot $P(\lambda,\varphi)$ at *(x,y)* However they tacitly assume that φ and λ are in radians, so you may have to adjust the scales for angles in degrees, multiplying either or both by $180/\pi$ to make the *x* and *y* scales the same at the equator. See the program listing for an example of this. To use degrees and logarithms base 10, note that $ln(x) \approx 2.3026*log_{10}(x)$, (I use $\approx$ to mean *"approximately equal to"*) and also $\pi/4 = 45^0$ giving:

$$y \approx 2.3026 \cdot log_{10}(tan(\phi/2+45^0))$$

Proof of the Mercator Formula with a little bit of Calculus thrown in

To prove the Mercator formula, we must show that $\int_0^{\phi} sec(\phi)\,d\phi = ln(tan(\phi/2+\pi/4))$

Using *f(x)* to represent some *function of x*, called *f* here, that is an *expression whose value is dependent on x* (anything else in the expression is assumed to have a constant value over the range considered), we can start with the *Fundamental Theorem of Calculus*, one form of which is:

$$\textbf{\textit{if}}\ y(x) = \int f(x)dx\ \ \textbf{\textit{then}}\ \ d(y(x))/dx = f(x)$$

Read this as: ***if*** *we integrate f(x) with respect to x to obtain y(x),* ***then*** *if we differentiate y(x) with respect to x we obtain the original f(x)"*. In other words, integration is the reverse of differentiation. Note that *y* is just another function of *x* related to *f(x)* by this formula, and of course we could employ any letters instead of *f, x and y*; these are what it is conventional to use.
A real mathematician would state all sorts of conditions for these rules to hold; I am sure they won't give us any problems here.
Although differentiation is a routine process – you simply follow the rules and get the answer – integration is a matter of ***guessing*** what function, will, when differentiated, give you the original function – that is, applying the fundamental theorem. Although there are a lot of tricks you can use to make the guesswork easier, it is usually not a straightforward process, but people have persevered at integrating $sec(\varphi)$ until they eventually came up with the formula above. But for us, the easier path is to differentiate $ln(tan(\varphi/2+\pi/4))$: if we get $sec(\varphi)$ then we have the proof.
In terms of functions, note that these are functions of φ only since $\pi/4$ is a constant, approximately *0.7854*, although it is customary to keep it in the exact $\pi/4$ form, and I will stick to that, and use $\pi/2$ later as well.
Formula (1) is a *function of a function* in the jargon, i.e. It is the *ln (a function)* of $tan(\varphi/2+\pi/4)$ which is *another function involving tan and* φ. We can write it as: $f(g(\varphi))$ where *f()* is *ln()*, and $g(\varphi)$ is $tan(\varphi/2+\pi/4)$. The rule for this is $d/dx(f(g(x))) = d/dx(f())d/dx(g(x))$, that is we differentiate the *f()* and multiply it by the differential of *g(x)*. Applying it to our formula, this can nest on to further levels:

$d/dx(f(g(h(x))))$, where in our case now $g()$ is *tan* and $h()$ is $(\varphi/2+\pi/4)$.
Because I am accustomed to this use of the letters, I will henceforth use U instead of h so that now $U(\varphi)=(\varphi/2+\pi/4)$. This makes the equations simpler, and typos less likely; so now we need to show that $d/d(\varphi)(ln(tan(U)))=sec(\varphi)$, i.e. *differentiating ln(tan(U)) with respect to* φ *gives the result* $sec(\varphi)$.
To do the differentiation we need some basic rules, and the first is for $ln()$, and it is $d/dx(ln(x)) = 1/x$. It follows from: If $x=e^y$ then $y=ln(x)$, which , as I described in appendix 1, is the *definition of ln(x)*. One definition of $e(..)$ itself is that it equals its differential: $\mathbf{d/dx(e^x)= e^x}$. So $d/dx(ln(x)) = dx/dy = d(e^y/dy = e^y = x$ giving $\mathbf{d/dx(ln(x))} = d/dx(y) = dy/dx = 1/(dx/dy)) = \mathbf{1/x}$
For the tan function there is a standard basic textbook rule: $d/dx(tan(x)) = sec^2(x)$. This is the form it is usually stated, but soon we will need $sec^2(x) = 1/(cos^2(x))$. I don't have the space to derive these here.
So now we have enough to start the differentiation, with a *function of a function* as above (I have put this in as a graphic in order better to show the structure of the formulae):

$$\frac{d}{d\phi}(ln(tan(U))) = \frac{1}{tan(U)} \cdot \frac{d}{d\phi}(tan(U)) = \frac{1}{tan(U)} \cdot sec^2(U) \cdot d\frac{(U)}{d\phi}$$

which simplifies a bit to

$$= \frac{cos(U)}{sin(U)} \cdot sec^2(U) \cdot \frac{d(U)}{d\phi} = \frac{cos(U)}{sin(U)} \cdot \frac{1}{cos^2(U)} \cdot d\frac{(U)}{d\phi} = \frac{1}{sin(U) \cdot cos(U)} \cdot \frac{d(U)}{d\phi}$$

Another standard trigonometric formula, which we will need in a moment, is: $sin(x+y) = sin(x)*cos(y) + cos(x)*sin(y)$.
From it we can easily derive a double angle formula by putting $y = x = U$ in this formula. We get $sin(2U) = 2*sin(U)*cos(U)$ and from this , by reversing and inverting it we have:

$$\frac{1}{sin(U) \cdot cos(U)} = \frac{2}{sin(2U)} \quad \text{so now} \quad \frac{d}{d\phi}(ln(tan(\phi/2+\pi/4))) = \frac{2}{sin(2U)} \cdot \frac{d(U)}{d\phi}$$

Expanding U in $sin(2U)$ gives $sin(2U) = sin(2(\varphi/2 + \pi/4)) = sin(\varphi + \pi/2)$ so, using the above formula for $sin(x+y)$ and substituting $x= \varphi$ and $y=\pi/2$, we get $sin(\varphi+\pi/2) = sin(\varphi)*cos(\pi/2) + cos(\varphi)*sin(\pi/2)$
Now $sin(\pi/2)=1$ and $cos(\pi/2)=0$, so: $\mathbf{sin(2U)} = sin(\varphi+\pi/2) = sin(\varphi)*0 + cos(\varphi)*1 = \mathbf{cos(\varphi)}$

Putting this together we have reached this point:

$$\frac{d}{d\phi}(ln(tan(U))) = \frac{1}{sin(U) \cdot cos(U)} \cdot \frac{d(U)}{d\phi} = \frac{2}{cos(\phi)} \cdot \frac{d(U)}{d\phi}$$

so we now need to differentiate the $\frac{d(U)}{d\phi}$ part (and I have changed the notation slightly):

$$d(U)/d\phi = d/d\phi(U) = d/d\phi(\phi/2+\pi/4)$$

There is another standard rule to cover this case of two functions, $\varphi/2$ and $\pi/4$, added together: The rule is that $d/dx(f(x)+g(x)) = d/dx(f(x)) + d/dx\ g(x))$ or: *to differentiate a function that is two further functions added together (or subtracted) we differentiate the two functions and add (or subtract) the results*, so $d/d(\varphi)(\varphi/2 + \pi/4) = d/d(\varphi)(\varphi/2) + d/d(\varphi)(\pi/4)$. Now the differential of a multiple of the dependent variable , in this case 1/2 of φ, is just the multiple itself, 1/2; and the differential of a constant such as pi/4 is zero; giving:

$$\mathbf{d(U)/d\phi} = d(\phi/2+\pi/4)/d\phi = d(\phi/2)/d\phi + d(\pi/4)/d\phi = 1/2+0 = \mathbf{1/2}$$

Putting all this in we have:

$$\frac{d}{d\phi}(ln(tan(\phi/2+\pi/4))) = \frac{d}{d\phi}(ln(tan(U))) = \frac{2}{cos(\phi)} \cdot 1/2 = \frac{1}{cos(\phi)}$$

$$= \mathbf{sec}\,\phi$$

.. so it's proved.

Editor's note: For technical reasons, which we discovered too late to change, the Greek letter "phi" appears as lower case in the text and upper case in the graphics throughout this article. Please accept out apologies for this.

```
100 REMark  Qtest_bas
110 :
120 CLS #0: INPUT #0\\"  Maps: 1 GB, 2 World (plain), 3  World
(Mercator):"!a$
130 REMark Check on a$ omitted
140 choice%=a$
150 COLOUR_PAL
160 CLOSE
170 ww%= FOPEN(con)
180 REMark Adjust the window size to suit
190 REMark your screen -- e.g. this works:
200 REMark  WINDOW #ww%,128,64,0,0: CLS#ww%
210 REMark But, for 1280 x 1024 screen
220   WINDOW #ww%,995,650,0,20: CLS#ww%
230 PAPER #ww%,1: INK #ww%,0: CLS #ww%
240 REMark Scales of maps are in degrees
250 IF choice%=1 THEN
260   REMark For GB: 15 high, bottom left at 10W, 50N
270   SCALE #ww%,15,-10,Merc(50)
280  ELSE
290    REMark 320 high allows for Merc expansion
300    REMark   beyond the undistorted 90S to 90N
310    REMark Bottom left at 180W, 150S
320    SCALE #ww%,320,-180,-150
330 END IF
340 asprat= .75
350 fc%=FOP_IN(win1_Worldcoast_coast0.lns)
360 IF fc%<0 THEN PRINT "NoGo": STOP
370 :
380 grid
390 Bearing
400 INK #ww%, 0
410 :
420 REPeat loop
430   IF EOF (#fc%) THEN PRINT #0\\"end": STOP
440   INPUT #fc%,x$
450   REMark Some lines (separators?) are:
460   IF '*' INSTR(x$) THEN NEXT loop
470   lat= x$
480   long= x$(' ' INSTR(x$) TO)
490   REMark The data file has oddities that spoil any
500   REMark attempt to plot the coast with lines thus:
510   REMark LINE#ww% TO asprat*long,lat
520   POINT#ww%,asprat*long,Merc(lat)
530 END REPeat loop
540 :
550 REMark End of main program ================================
560 :
570 DEFine FuNction Merc(x)
580   REMark Work in radians, but lat long etc in degrees
590   REMark PRINT Lct%,x
600   IF x==0 THEN RETurn 0
610   IF choice%= 2 THEN RETurn x
620   RETurn (180/PI)*LN(TAN((x*PI)/360+PI/4))
630 END DEFine Merc
```

```
640 :
650 DEFine PROCedure grid
660   LOCal i, bot, top
670   REMark Not plotted for GB map
680   IF choice% = 1 THEN RETurn
690   :
700   REMark Latitudes at 15 degree intervals
710   REMark Emphasise Equator and 90 degree meridians
720   IF choice% = 2 THEN
730     REMark Whole world map on recangular grid
740     bot= -90: top= 90
750    ELSE
760     REMark For Mercator avoid infinities at poles
770     bot= -75: top= 75
780   END IF
790   FOR i = bot TO top STEP 15
800     IF i==0 THEN INK#ww%,9: ELSE INK#ww%,11
810     LINE #ww%, -180*asprat,Merc(i) TO +180*asprat, Merc(i)
820   END FOR i
830   :
840   REMark Meridians at 15 degree intervals
850   REMark same on all maps
860   bot= Merc(-89): top= Merc(+89)
870   IF choice% = 2 THEN
880     bot= -90: top= +90
890   END IF
900   FOR i = -180 TO +180 STEP 15
910     IF i==0 OR ABS(i)==90 THEN INK#ww%,9: ELSE INK#ww%,11
920     LINE #ww%, asprat*i,bot TO asprat*i,top
930   END FOR i
940   REMark  FOR j = -180 TO + 180 STEP 15
950 END DEFine
960 :
970 DEFine PROCedure Bearing
980   LOCal x,xx
990   IF choice% =1 THEN RETurn
1000   IF choice% =2 THEN
1010     FOR x= -85 TO 85 STEP .1
1020       xx= (180/PI)*LN(TAN((x*PI)/360+PI/4))
1030       POINT#ww%,asprat*xx,x
1040       REMark POINT#ww%,-asprat*xx,-x
1050     END FOR x
1060   END IF
1070   IF choice%=3 THEN
1080     FOR x= 0 TO 150 STEP .1
1090       POINT#ww%,asprat*x,x
1100       POINT#ww%,asprat*-x,-x
1110     END FOR x
1120   END IF
1130 END DEFine
1140 :
1150 DEFine PROCedure backup
1160   SAVE win1_WorldCoast_Qtest_bas
1170 END DEFine
1180 :
```

Steve Poole writes:
In your last QL Today article 'End of the Season', you asked if anyone could write a program to calculate a person's Carbon Footprint.
This program would have to be very long, as it would have to calculate the total footprints of all the goods and services you consume, with each one assessed from the extraction of the raw materials to the transformation, transport and recycling overheads both for currently consumed articles and also to fixed assets such as old houses, holiday homes etc, right down to the environmental costs of your birth, funeral and health expenses.
The simple solution is to let experts do the work for you.
Just look up Carbon Footprint Calculator on Google, or better if you speak french, ADEME for a highly detailed breakdown showing you how to drastically reduce the carbon value of whatever you do in life.
But in a nutshell, most scientists agree that we should divide our energy consumption by four within the next few years and switch to sustainable energy sources. We can then slowly eliminate greenhouse gases by converting trees into organic fertiliser. This is fully realistic, only the oil company shareholders and governments do not promote it, because of the vast amounts of revenue they make from high energy use. But there is no alternative: either we adapt to the limitations imposed by planetary resources or we destroy most life forms on earth.

Colin Mckay writes:
To assist people obtain Building Warrants I calculate wind loading on buildings. It is a tiresome job, & I have programmed, on the QL, the calculation as far as calculating the wind pressure. Three topographical factors are needed:
The orientation of the building, because characteristic wind speeds vary with direction. The elevation of the site, because wind speed varies with height above Mean Sea Level. The roughness of the terrain crossed by the approaching wind, principally whether open country or town.
There is a possible fourth which is the variation of the ground immediately windward of the site, e.g. escarpment. Not required so often.
The most awkward item to obtain is the height of the site. When this is located on the 1" OS map that I possess, this can help, but not where the site is in a built-up area. Otherwise I use Multimap on the Internet. It would save time if there were a topographical model. These do exist in commercial programs which do the entire wind loading calculation. It is less costly to use the QL, but there is no topographical model to use on it.
When the architect provides an adequate site plan, and a post code, the orientation of the building can be obtained again from Multimap, or Streetmap.
Wind loading obviously cannot be accurately calculated, but the game is to present the most plausible numbers which satisfy the design code (BS6399:Part 3) and the circumstances.
I believe that at least one commercial program gives heights based on averages over 1 km squares. A trifle coarse, but better than nothing.
Possibly you have not realised that the modern timber frame houses have to be designed to resist wind loading. Another Engineer told me that when the wind is strong and in a particular direction there are creaking noises from his house, which he attributes to the wind on the roof deflecting the timber walls, which causes the roof to rub against the brickwork. There ought to be a gap between the top of the brickwork and the underside of the roof eaves, and he presumes that there is no gap.
A QL topographical model would be useful.
ED: Colin McKay sent this to me as a personal email, but agreed to its publication in QL Today. Although Colin thought it too specialist for inclusion in the magazine. I felt it was an interesting example of a professional use of the QL. Our GPS and mapping articles have interested a number of readers and we would be pleased to hear from other readers about their interest in mapping software or other professional use of the QL.
In a subsequent email Colin wrote that he would like to see Wind Speed maps reproduced on the QL:
"The wind loading calculation starts off with the value of the basic wind speed at the site. The Engineer just looks at the map, and decides by eye where the site is on the map, then reads the speed to one decimal place, as assessed by eye.
Probably in the commercial wind loading programs the Engineer enters the Grid

Reference of the site, and without even seeing the map, the speed is displayed on his screen, and the value is fed into the calculations. The value may even be to an accuracy of two decimal places, but the accuracy is spurious.
The aspect which I am chasing is convenience, not accuracy, because to complete the loading calculation often 'assessments' have to be made, which are little better than guesses. E.g. a simple rectangular plan building with a simple end-to-end ridged roof conforms to the data provided in the BS, but when dormer windows and towers are added to the roof profile who knows what pressures and suctions the wind creates? The Engineer just adopts a value from his calculations, which ought to give him reasonable protection against being sued.
My method of working is that the construction drawings appear on the PC, and the computer calculations appear on a QL. Although I have QPC2 I prefer to have the drawing available whilst I calculate, and I would have the wind map on the QL, either as a separate program, or as part of a wind loading program. Separate seems to be preferable, because large means complication, and sometimes groping around a long listing, which is time consuming."

Norman Dunbar writes:
Subject: Response to Wolfgang's article on using QPC under Linux (Vol 11, issue 4)

Hi Geoff,
I was wondering to myself about getting QPC to run on Linux and remembered that I had read an article in QL Today so I hunted in the as yet unpacked boxes and found Volume 11 Issue 4 where Wolfgang Lenerz has an article on this very subject.
Reading through it I noticed that Wolfgang seems to be having a lot of problems with Linux's habit of considering upper and lower case letters to be different, even in file names.
As a Suse user of some time, I know an easy way around Wolfgang's problems, as follows:
in the command line (shell) start typing the first few letters of the file name, then press TAB. The shell (Bash in my case) will auto-complete as much of the remainder of the file name as possible. If there are two or more files with the same start to their name, they will then be listed and you can type one or more characters to make the name unique again, then press TAB again. Continue in this manner until the full file name has been entered.

So, on page 15, where Wolfgang is explaining how he had to run the 'rpm' command and type the full name of his rpm file correctly, he could have made life so much easier as follows:
`rpm -Uvh wine‹TAB›`
This way, unless there were more files in the /tmp directory, where he was working at the time, with names beginning with the word 'wine', the shell would have typed in the remainder of the file name and Wolfgang wouldn't have had to be careful about getting each and every letter in the correct case.
Additionally, he explains how he had to install (or unpack) the rpm file by running a shell and becoming root etc etc. Too much work there I'm afraid. There is a much easier way than this.
In Konqueror, the file browser, navigate to where the rpm file has been saved after download, and click it (ok, you may have to double click - it depends on how you have set up your system). Konqueror will open a page giving details of the purpose of the rpm file. At the top is a button with the legend 'Install package with YAST' - click this.
After a small delay, you are prompted for the root password. Enter the root password and YAST will open up and install the rpm file. No messing about with shells, becoming root and so on.
It has to be said however that running the 'rpm' command as described by Wolfgang is much quicker then using YAST as YAST goes off and makes sure that all its repositories are up to date and so on.
While I'm not a Linux expert, it does have many ways to make life easier - especially when typing in a file name.
The auto-completion of file names works down a directory structure as well, so to type in the file name
`'/home/norman/downloads/OracleXE/oracle-xe-10.2.0.1-1.0.i386.rpm'` for example - say to install it, I would, as root, type the following : `rpm -ivh /ho‹TAB› n‹TAB› dow‹TAB› Ora‹TAB› ora‹TAB›`
Each time I hit the TAB key, the shell will fill in the remainder of the file name as far as it can at that point. SO the above combination of keys would give the following results :
`/ho‹TAB›` gives `/home/`
`n‹TAB›` gives `/home/norman/`
`dow‹TAB›` gives `/home/norman/downloads/`
`Ora‹TAB›` gives
`/home/norman/downloads/OracleXE/`
`ora‹TAB›` gives
`/home/norman/downloads/OracleXE/oracle-xe-10.2.0.1-1.0.i386.rpm.`

I can use as many letters as I like before pressing TAB, the more unique I make it, the less chance there is of me having to try again. If, when I pressed TAB, the shell responded with nothing, then at that point I know I have typed a wrong character in the filename, and can correct it (with backspace) and type the correct one.

Now, hopefully, I can get on and install wine and QPC under SuSE 10.3 and make it work. I'm disappointed by Wolfgang's findings that QPC runs extremely slowly under Linux and will be looking into this when I get things sorted out.

Improving QL Emulator I/O (Input/Output) Capability

by Ian Burkinshaw

We are supposed to be tinkerers, and having seen Hugh Rooms articles on GPS (QL Today Volume 11, issues 2,3 and 4) which involves external equipment in the form of the GPS receiver. I have been playing with Hugh's program with two types of GPS receiver, and I am planning to do a follow up article of my experiences on this subject. However I thought it may be a good idea to look at other forms of interfaces that we could use that may have a wider interest. Do let QL Today or myself know if these technical hardware/software projects are of interest. Even if you don't build/use them they may still be of interest.

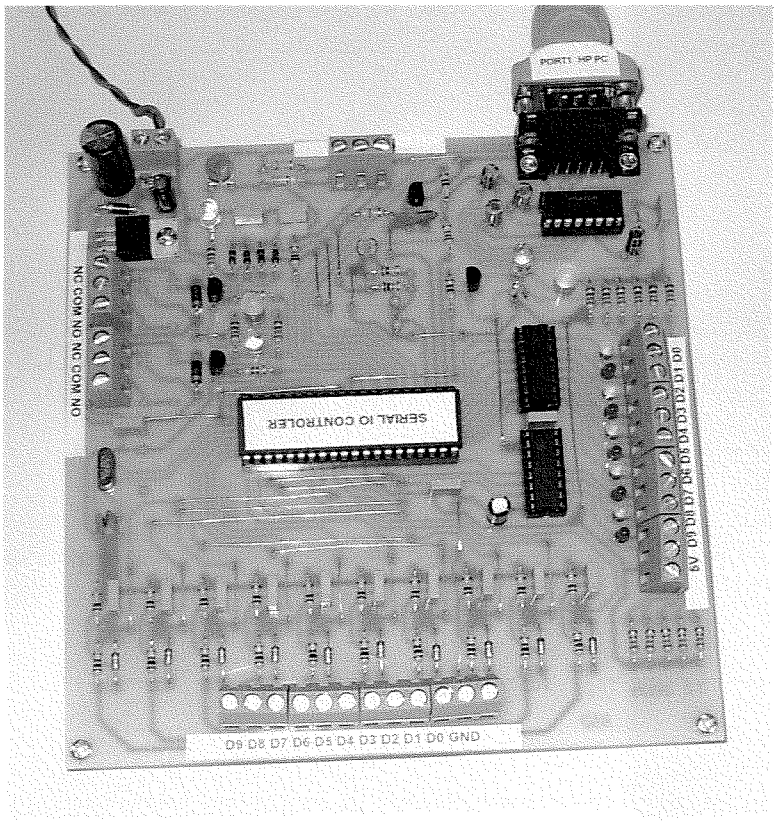
In the last few months there have been two projects published in Everyday Practical Electronics magazine which could be of interest. One being an oscilloscope project which would also make a fairly fast, by QL standards, (up to 40Khz) analogue to digital converter, it also has two channels. So could be used for stereo audio applications. This project is published in two parts, in the August and September 2007 issues of the magazine. The other project of use is the Serial I/O Controller project in the Jan 2008 issue of said magazine. In both cases these projects use the standard RS232 interface to the computer.

I have chosen the Serial I/O Controller project first since it is the simplest of the two projects. I will cover the oscilloscope in a future article. The other reason for going with the Serial I/O Controller is it will work with all QL systems, since it communicates at a low baud rate of 2400 Baud. The oscilloscope project needs higher than 9600 Baud rate which not all QL system can support. The original article is not totally correct in saying it will work at 9600 Baud rate, more on that when I cover that project.

If you can solder then you can make this project. The PCB and the components are easily available. The PCB from Everyday Practical Electronics(1) publishers and most of the components from Farnell(2) for example, but there are other places such as RS, Rapid etc. If you do not have the ability to program a PIC (Microcontroller chip), then pre programed PIC's can be obtained from Magenta Electronics(3), who advertise in the magazine. The cost of the board project is about £50.00 this will vary depending were you purchase, so please do not hold me to this, it is just a guide. You will also need to add some form of power supply, the board needs between 9-15VDC with 100mA capability should be more than enough, my test board takes 25mA at idle, with all the LED's lit and relays closed it rises to about 50mA, so one of these wall plug in low voltage type units would be good and safe. There is a safety diode on the supply input, so if you do connect to the power input incorrectly you will not do any damage. Also you will need a 9 way D Type serial cable, neither of these items is included in my guide price for the project. You do not need many tools either, side cutters to cut the component leads, small pliers to bend component leads, soldering iron 15W, solder, good light, and magnifying glass so you can check all your soldered joints and also check for any shorts across PCB tracks. Also a small screw driver. A cheap digital multimeter, some are as low as £5 would also be useful. For example build the board, but do not insert the PIC, or other IC's except the input voltage regulator (REG1), power up the card, the LED (Light Emitting Diode) LED4 should light up, if not disconnect the power and check for shorts and the polarity of the LED. With the multimeter you can check the voltage output from REG1 is 5V. If the LED is lit and you have 5V things are looking very good. The instructions in the original EPE article are fairly good, I built mine from the instructions to make sure they were accurate for this article, and did not find any problems and my

board worked first time. However I must come clean and say I am an electronics engineer by training so I do have an advantage, but I tried to look at this as if an inexperienced constructor. They even give you the resistor colour codes to follow. The most important diagram is the one on page 18 which gives you all the component positions and orientations, components you need to watch for in this regard are some capacitors, diodes, transistors, LED's (note the short leg on an LED's is K(Cathode), and the IC's(Chips), watch for the notches or dot's, the pin next to a dot is pin 1, which will be the same end as a notch. You will also need some 24SWG tinned copper wire for the wire links on the board of which there are 34. I strongly advise that you use IC sockets, there are 4 required, 1 40 pin and 3 16 pin. Again watch the orientation, there is notch at one end of these and this should line up with the IC(chip) notch, they are clearly shown in the EPE diagram. See my picture below this may help. Please note I have put an LED in place of the Buzzer for testing purposes, the Buzzer was driving me round the bend while testing. Also, I have at the stage this picture was taken not fitted the relays either. As you can see the board does work, this picture was taken with the board powered and being driven from my test program.

CAUTION: THIS BOARD IS NOT MAINS RATED, SO DO NOT CONNECT ANY MAINS VOLTAGES DIRECTLY TO THE BOARD, EVEN THE RELAYS. MAINS CAN AND DOES KILL. YOU HAVE BEEN WARNED.



All the required PIC software and PC Visual Basic based software is available free of charge from the EPE web site. Also the source code is included so you see how the project works. Please note you may need the SerialOCX software also downloadable from the EPE web site, for the Visual Basic applications to work on your PC. This is due to the way some of Microsoft systems work, in some cases access is needed to be given to the serial hardware within the PC. If you are not going to use the Visual Basic software then it is not required.

With the QL hardware, there have been over the years a number of options for getting external signals in and out of the QL. The Minerva, I think was the best, with the I2C serial communication capability. These were and as far as I know still are available from Tony Firshman of TF Services(4), I still have and use mine. Producing extra inputs or outputs just means adding the appropriate chip (IC). In fact up 256 I/O chips could be used on one serial line. Each chip having it's own address, selectable by setting pins high or low on the chips themselves. These I/O chips come in various types. The most useful to us are the parallel digital(PCF8574) and analogue (PCF8591) chips. TF Services supply pre built units in their own cases and DIP switches for setting the address. The only disadvantage is you cannot run I2C over long distances, a metre or so is the limit. But we lost the ability to use I2C with emulator based systems. However all is not lost. The Serial I/O Controller gives us a 10 bit parallel input and 10 bit parallel output, 4 A/D(analogue to digital) converters, of which one is connected to an LDR (Light Dependant Resistor), the second to a LM335Z temperature sensor. The third input can measure voltages from 0 to 25 volts DC and the fourth measures voltages from 0 to 5 volts DC. However these can be changed to take other inputs with simple additions and minor hardware changes, like, for example, a pressure sensor. So one has the makings of a weather station.

If you need more capability than one board can supply, then you can use more than one board, it just depends how many RS232 serial ports you have on your system. A lot of PC's these days do not have any RS232 ports, however you can get fairly cheaply USB to RS232 converters, there are also parallel ones available

as well if you only require 8 parallel outputs. QPC for example can support up to 8 serial ports. So you could have a system with up to 80 parallel input lines, 80 parallel output lines, 16 relays, 8 light sensors and 8 temperature sensors and 16 analogue inputs using 8 Serial I/O Controller boards. Not as many as a I2C based system but I think this would satisfy most people. Also the oscilloscope project that can be used on another RS232 port at the same time as the Serial I/O Card, the limitation being the 8 RS232 ports.

On the Serial I/O Controller there is also an on board buzzer and two relays. With the temperature sensor and the relays we have the making of a thermostat type control system. Using the QL's internal clock we can combine this with a time schedule. We can also log temperature and light information against time and create a log to analyse later. This is using the board as designed, no changes required to the published hardware.

In the original EPE article details are given as to the serial protocol to drive the board. With this and the listing below you should be able to get things going quickly. The list gives you procedures for accessing all the features of the board, I/O, temperature, light (LDR), relays, buzzer and the A/D converters.

```
100 REMark saved as win5_SerialCon_test
110 REMark Serial IO Controller test program V1, by Ian Burkinshaw 6 Jan 2008
120 BAUD 2400
130 OPEN#5;ser1ir
140 OVER 0:CLS:CSIZE 1,3
150 FOR c%=0 TO 1023
160 read_ldr
170 read_temp
180 read_ana1
190 read_ana2
200 read_10bit
210 set_output c%
220 display_data
230 drive_relay 1,0
240 drive_relay 2,0
250 drive_relay 1,1
260 drive_relay 2,1
270 drive_buzzer 0
280 drive_buzzer 1
290 NEXT c%
300 CLOSE#5:CSIZE 0,0:STOP
900 REMark ************************************************
1000 DEFine PROCedure set_output(cc%)
1010 c1%=0:c2%=0
1020 a$="a"&CHR$(8)
1030 PRINT#5;a$;
1040 REMark delay before sending next command
1050 PAUSE 2
1060 a$="a"&CHR$(cc%)
1070 PRINT#5;a$;
1080 REMark delay before sending next command
1090 PAUSE 1
1100 REMark Routine for bits 9 and 10
1110 IF cc%>=256 THEN c1%=1
1120 IF cc%>=512 AND cc%<=767 THEN c1%=0
1130 IF cc%>=512 THEN c2%=1
1140 a$="a"&CHR$(c1%)&CHR$(c2%)
1150 PRINT#5;a$;
1160 END DEFine set_output
1170 REMark ************************************************
1180 DEFine PROCedure read_temp
1190 a$="a"&CHR$(1)
1200 PRINT#5;a$;
1210 temp$=""
1220 read_data
1230 temp$=a1$
1240 END DEFine read_temp
```

```
1250 REMark ************************************************
1260 DEFine PROCedure read_ldr
1270 a$="a"&CHR$(2)
1280 PRINT#5;a$;
1290 ldr$=""
1300 read_data
1310 ldr$=a1$
1320 END DEFine read_ldr
1330 REMark ************************************************
1340 DEFine PROCedure read_ana1
1350 a$="a"&CHR$(4)
1360 PRINT#5;a$;
1370 ana1$=""
1380 read_data
1390 ana1$=a1$
1400 END DEFine read_ana1
1410 REMark ************************************************
1420 DEFine PROCedure read_ana2
1430 a$="a"&CHR$(3)
1440 PRINT#5;a$;
1450 ana2$=""
1460 read_data
1470 ana2$=a1$
1480 END DEFine read_ana2
1490 REMark ************************************************
1500 DEFine PROCedure read_10bit
1510 a$="a"&CHR$(7)
1520 PRINT#5;a$;
1530 bit$=""
1540 read_data
1550 bit$=a1$
1560 END DEFine read_10bit
1570 REMark ************************************************
1580 DEFine PROCedure read_data
1590 REMark first syncronise to syncronisation character 'a', also will keep runing
     loop until character appears, so no PAUSE command required wait for the PIC to do
     it's job.
1600 REPeat loop1
1610 a$=INKEY$(#5)
1620 IF a$="a" THEN EXIT loop1
1630 END REPeat loop1
1640 a1$=""
1650 REMark extract data until termination character '@'
1660 REPeat loop2
1670 a$=INKEY$(#5)
1680 IF a$="@" THEN EXIT loop2
1690 a1$=a1$&a$
1700 END REPeat loop2
1710 END DEFine read_data
1720 REMark ************************************************
1730 DEFine PROCedure display_data
1740 AT 0,0:PRINT "LDR Data : ";ldr$;"   "
1750 AT 1,0:PRINT "Temp Data : ";temp$;"   "
1760 REMark Next line displays the data and voltage conversion for analogue input 1,
     reads 0 to 5 VDC
1770 AT 2,0:PRINT "Analogue Data 1 : ";ana1$;"   ";ana1$/204.6;" Volts                "
1780 REMark Next line displays the data and voltage conversion for analogue input 2,
     reads 0 to 25 VDC
1790 AT 3,0:PRINT "Analogue Data 2 : ";ana2$;"   ";ana2$/40.92;" Volts                "
1800 AT 4,0:PRINT "10 Bit Input Data : ";bit$;"   "
1810 AT 5,0:PRINT "10 Bit Output Data : ";c%;"   "
1820 END DEFine display_data
1830 REMark ************************************************
```

```
1840 DEFine PROCedure drive_relay(rn,nf)
1850 a$="a"&CHR$(5)
1860 PRINT#5;a$;
1870 PAUSE 2
1880 a$="a"&CHR$(rn)&CHR$(nf)
1890 PRINT#5;a$;
1900 END DEFine drive_relay
1910 DEFine PROCedure drive_buzzer (nf)
1920 a$="a"&CHR$(6)
1930 PRINT#5;a$;
1940 PAUSE 2
1950 a$="a"&CHR$(nf)
1960 PRINT#5;a$;
1970 END DEFine drive_buzzer
```

You will see from the listing that there are PAUSE statements, these are very important. Since after sending a command to the board the PIC has to process the command before it can read the next command or command data, so you have to give the board time to do this. The PAUSE command works in frames per second, in our case here in Europe there 50 frames per second. Or put another way PAUSE 1 equals 40ms, the processing time for most commands is about 50 ms, so PAUSE 2 is about right. It does not matter if the delay is longer, only if it is too short problems occur. The other issue to take into account is that RS232 communication supports only 8 bits at a time, not ten. So for example setting the parallel outputs requires you to send one word, to represent the first 8 bits then two further words that are either 1 or 0 as required to represent bits 9 and 10. Look at the procedure SET_OUTPUT, variable c1% is bit 9 and c2% is bit 10. The IF statements then sort out when they should be set, depending on the input variable cc%. To use this procedure, 'SET_OUTPUT [variable]', the variable can be between 0 and 1023. Note there is no error trapping if you exceed this value. If you run the program as is you should see the Serial IO Controllers output LED's count in a binary fashion. Lines 100 to 290 with procedure 'DISPLAY_DATA' are just to test and show basic operation for testing and as an example. Reading data back from the board, the procedure 'READ_DATA' looks for the synchronising character after the required delay from sending a command to then return data, the character which is 'a' ASCII code 97. Then the data value itself is sent. The data stream is terminated with the character ':' ASCII code 64.

Hope that gives you some ideas and that if you are brave enough to try it, that you have fun. It may make a good school project as well. May be you could write an article for QL Today about how you use this project. Now let me get the oscilloscope card going.

Sources and References

I have no connection with any of the organizations below. They are just what I have used in the preparation of this article.

(1) "Everyday Practical Electronics" magazine
http://www.epemag.co.uk

(2) "Farnell" source of electronic components
http://www.farnell.com

(3) "Magenta Electronics" source of pre-programmed PIC's
http://www.magenta2000.co.uk

(4) "TF Services" source of Minerva and I2C products.
http://www.firshman.co.uk

Off-Screen Drawing

by Steve Poole

When the QL appeared in 1984, it had a revolutionary graphics system for the time. Firstly, it could work on a theoretical window of 1E615 scale units high. That was far more than enough for most scientific calculations! Secondly, it could work on that 1E615 window outside of the 512*256 pixel screen. This meant that you could drive huge plotting tables in great detail, without having to modify the screen view all the time.

This is because SuperBasic does not test to see if drawing is starting to go off-screen, but that means you may have to wait a long time to see if there is any output at all. So I decided to bench-mark off-screen drawing to see how it compares to on-screen drawing. To do that I decided to test within exact 2-power integer limits, to avoid any floating-point coercion overheads. Furthermore I decided to see if filling

affected the results. To see what I found, just run the listing.
With filling turned on, the time taken to completion varies from less than 1 second to 30 seconds, depending on which percentage of the square is off-screen. (On my machine, I am running 1000 loops on QPC at 2.8 Ghz). For some reason, some squares are outlined, but not filled at all... With filling turned off, those 30 seconds are only reduced to 21. But with a different off-screen percentage, a 6 second delay is reduced to 1, a far greater time saving. This suggests that the FILL algorithm is quirky.
Surprisingly, with side 'n' and scale 'sc' set at 1E610, the maximum delay with filling is but 6 seconds. This result was most unexpected, as one would have assumed that more time would be taken with greater LINE distances to cover.
It seemed evident that in the worst case scenario, it would be advantageous to include a test to see if the line was going off-screen and to limit drawing to on-screen only. This was achieved by the IF TEST structure, which restricts the boundaries in an optimal manner. But to no avail: The 30 second delay remains 30 seconds, meaning that you cannot improve such timings by adding SuperBasic code. Perhaps a little machine-code might help?
Therefore, in conclusion, calculations are mainly dependant on the vertical Scale factor, but I cannot uncover any logical explanation as to why there are such huge variations. But now at least you see that you can expect to wait a long time under certain configurations...So select your parameters carefully, and work by trial and error when you write your code if you wish to optimise it.
Please feel free to experiment with the listing. If you discover the logic behind these variations in timing, please contact the Editor so as to help other QL programmers to fine-tune their drawing code. There could be very important time savings to be made...

```
100 ::
110 REMark Off_screen_bas, by S.Poole. v3feb2008.
120 :
130 CLEAR: WINDOW 256,256,256,0: CLS#0: CLS: CLS#2
140  REMark n=side length: fc=factor.
150  n=32767: sc=256: ik=0: fl=1: test=0
160   FOR fc=1 TO 5,10 TO 100 STEP 25
170       FOR scl=sc*fc: PRINT#2,scl,TIME(scl)
180   END FOR fc: WINDOW 256,206,0,0: STOP
190 :
200 DEFine FuNction TIME(sca)
210  start=DATE: ik=ik+2: INK ik
220  IF start=DATE: GO TO 220: ELSE start=DATE
230  SCALE sca,0,0
240  :
250  REMark Only loop 50 or more times on slow machines:
260  FOR loop=1 TO 1000
270      IF test: IF sc>256: n=sc
280      FILL fl: LINE 0,0 TO 0,n TO n,n TO n,0 TO 0,0: FILL 0
290  END FOR loop: RETurn DATE-start
300 END DEFine
310 ::
```

You are getting closer towards the end of this issue! Please take a few moments to fill in the Renewal Form which came with this issue and return it to J-M-S or QBranch.
Every reminder we do not have to send saves costs, reducing the need to adjust the price in the future. Thanks a lot!

RWAP Services NEWS!

**** We have moved ****

See our updated address details below.

We have also acquired more brand new Sinclair QL membranes and another stock of Epson Stylus Colour 850 inkjet printers, so if you need a better printer for your QL, give us a shout.

More news is always available on our website: www.rwapsoftware.co.uk

We are also looking to produce some new hard disk interfaces for the ZX Spectrum and have a few little projects on the drawing board.

Our websites:
http://www.rwapservices.co.uk (General site)
http://www.rwapsoftware.co.uk (Sinclair computer second hand and new items)
http://www.rwapadventures.com (Adventure Programs)
http://www.internetbusinessangels.com (Guidance on setting up online businesses).

New Products!

Q-WORD 1.0 NOW WITH DIGITAL SOUND ON QPC2!

The wait is now over! Q-Word version 1 is finally available!

Platforms:
QPC/QXL, Q40/Q60, Aurora (with SGC)

Prices:

All versions without P-Word	£20.00
All versions with P-Word	£30.00

Notes:
Q-Word DOES NOT require SMSQ/E with GD2 support -OR- SMSQ/E at all on the Aurora or Qx0 machines. It works on the highest colour depth everywhere regardless of Operating System.
The Aurora version is available on either HD or ED disk. For the latter add £1.00 to the price. ED version is uncompressed and can be run directly from the floppy. All other Floppy versions are compressed. QPC/QXL version comes on CD. Non CD versions DO NOW support digital sound on QPC2

TALENT COMPUTER SYSTEMS for **Windows**

For QLers that run Windows or with incompatible hardware for Talent Games, we now have re-released these adventures so that they can run on your Windows-equipped PC. No Emulator, floppies, microdrive backups etc. required, just a one-click install! Of course the full QL line is still available! (See side column)

Talent Games for Windows ea. £ 10.00
(Each Game includes a runtime installation of QLAY-2 by Jimmy Montesinos)

Games Currently Available from www.rwapadventures.com

The Lost Kingdom of Zkul
West
The Prawn
Return to Eden

Replacement Sinclair QL Keyboard Membranes

We always have a stock of brand new Keyboard Membranes (and keyboard parts) for the original Sinclair QL, so if you have some keys which no longer work, just give us a call.

Cost is only £18.50 plus £2.75 post and packing.

Second Hand Items - Huge Range Available

We stock a wide range of books, hardware and software for the Sinclair QL, Z88 and ZX Spectrum, including disk interfaces, memory expansion and microdrive cartridges. If there is anything you need - have a look at www.rwapsoftware.co.uk (or ring us with details of your requirements).

We are always happy to help.

Old Favourites!

Utilities

SBASIC / SuperBASIC Reference Manual on CD	£ 20.00
Sidewriter v1.08 *Landscape Printing (EPSON printers)*	£ 10.00
ImageD v1.03 *3D object generator*	£ 10.00
Q-Help v1.06 *Superbasic On-Screen help system*	£ 10.00
Q-Index v1.05 *Keyword-to-topic finder*	£ 5.00
ProForma ESC/P2 Drivers v1.04 for ProWeSs *Printer Driver*	£ 8.00

Applications

Flashback SE v2.03 (upgrade only) *Database*	£ 5.00
QL Cash Trader v3.7 *Accounting/Finance*	£ 5.00
QL Payroll v3.5 *Accounting/Finance*	£ 5.00
QL Genealogist v3.26 *Genealogy*	£ 20.00
Genealogy for Windows	£ 50.00
QL Genealogist to Windows version upgrade	£ 25.00
QL Cosmos v2.04 *Planetarium*	£ 5.00
Q-Route v2.00 *Route Finding*	£ 25.00
Upgrade from v1.xx	£ 5.00
Britain map v1.11	£ 2.00
BIG Britain map (needs 2Mb) v2.03	£ 5.00
Various Britain Area maps (ask for details)	ea. £ 2.00
Ireland map v1.00	£ 5.00
Belgium map v1.01	£ 2.00
Catalonia map v1.03	£ 2.00
P-Word UK English Dictionary (500.000 words!) *Dictionary*	£ 15.00

Leisure

Return to Eden v3.08 *Adventure*	£ 10.00
Nemesis MkII v2.03 *Adventure*	£ 8.00
The Prawn v2.01 *Adventure*	£ 8.00
Horrorday v3.1 *Adventure*	£ 8.00
West v2.00 *Adventure*	£ 5.00
The Lost Kingdom of Zkul v2.01 *Adventure*	£ 5.00
All 6 games above	£ 25.00
D-Day MkII v3.04 *Strategy/War Simulation*	£ 10.00
Grey Wolf v1.08 *Graphical Submarine Simulation*	£ 8.00
War in the East MkII v1.24 (upgrade only) *Strategy/War Simulation*	£ 5.00
Open Golf v5.20 *Sports Simulation*	£ 8.00
QuizMaster II v2.07 *Quiz*	£ 5.00
Stone Raider II v2.00 *Arcade Game*	£ 5.00
Hoverzone v1.2 *Arcade Game*	£ 5.00
Deathstrike v1.5 *Arcade Game*	£ 5.00
Flightdeck v1.0 *Flight Simulation*	£ 10.00
All 6 games above (Open Golf, QuizMaster II,Stone Raider II, Hoverzone, Deathstrike and Flightdeck)	£ 28.00

Notes on Software requirements
The following programs have a minimum SGC card requirement: P-Word, Qword, Big Britain MAP for Q-Route

RWAP Services
3 Dale View Court, Fulford, Stoke-On-Trent, Staffordshire ST11 9BA TEL: (+44) 1782 398143 From the UK Dial: 01782 398143
Website: http://www.rwapsoftware.co.uk
Email: sales@rwapsoftware.co.uk

We Accept Payment using:

PayPal VERIFIED NOCHEX PREFERRED PAYMENT (Cheques in £ sterling made payable to R. Mellor)

Byts of Wood

by Roy Wood

One of the big problems with having the kind of small user base the QL and related systems have is that, when people come up with good ideas, there is not a big machine behind them to push it forward. When I was younger I was always sure that someone was stealing my ideas but, I realise now, that a good idea will probably be had by several people at the same time, all over the world and that they will either fail to act on it swiftly enough, get something very wrong in its execution or or just decide that it is not feasible or viable and abandon it thus leaving the field open for one of the others who has the right combination of ideas, drive and muscle. This may be why so many things come out half baked or just plain wrong. It is the same in software and in hardware and it does not, of course, stop at the door of the computer world. There is a famous story (no doubt apocryphal) of the man who invented the dusting machine which would blow the dust off things. One of his peers in the audience took the tube from the machine and explained to him that all he was doing was blowing the dust around the room. He then proceeded to place the tube in his mouth and suck, removing the dust, inventing the vacuum cleaner and - incidentally - almost choking himself to death in the process.

In the course of its long life the QL community has thrown up a few examples of this process in action. the RomDisq, for example, came out long before the, now ubiquitous, USB stick made its appearance and Tony's Compuswitch was around for at least a year before the first commercial models arrived. There are probably more examples of this but I will move on to the main point which is the EEEPC.

EE its an EEEPC

Just before Christmas Asus announced its EEEPC. In case you do not know about this device it is a small laptop computer - well halfway between a laptop and a PDA actually. The display is 7 inch (178 mm), measured diagonally, and it has a resolution of 800x480 pixels. It has no hard drive, CD or Floppy but comes with a variety of Flash Ram sizes from 2Gb to 8Gb. It features all of the usual PC connections - well the modern ones anyway - such as USB, Ethernet, Wireless LAN and Audio and is supplied running LINUX. Windows XP drivers are also provided although, by the time this is loaded you would not have a lot of room for applications except on the larger storage versions.

You may wonder why I am reading out this advert for Asus here. Well there are two main reasons. Firstly, the whole idea of a small unit with a reasonably sized screen, using flash RAM in place of Hard drives was put forward by our very own Nasta some 3 years ago. It was an idea that excited a lot of people in the QL world because it was planned as a pure QL platform but, as far as I know, it never came to anything. The second reason for mentioning it here is that there was a discussion on the user group about running the QL emulation on this device. some wondered if QPC2 could be adapted to run under LINUX on it for instance and Norman Dunbar even said he would look into the possibility since Marcel did not have the time to do it. There are, of course,

QL emulations for LINUX and I believe that a couple of people have run these very successfully on the machine. I have never used any of the other, available, QL-emulations, having always been more than happy to run QPC2 so I am not sure of the comparative merits of each system but whatever emulation you choose to run on this it does do what many people wanted years ago and provide you with complete QL portability. This does, in fact, give you a, rather neat, QL in your pocket.

One trick that many of the people who discussed this seemed to miss (unless I am being woefully ignorant or optimistic here - wouldn't be the first time) is maybe to run DOS on the machine and then use QPC1. The machines have Celeron Processors so they should, in theory, be able to run DOS and then you could have an entire QL-laptop with nothing to get in the way. Whilst I realise that Marcel is now no longer a student and unable to devote as much time to the QL as he used to, maybe someone could take over and develop QPC1 for just this kind of situation. It would certainly be a very elegant thing and the screen size is more than adequate for a basic QL operation - shame it is not slightly higher but you cannot have everything can you?

Nagging

One advantage of being stuck in the past world of computing is that we don't quite have the same degree of child safety locks on our software. In the days when we started out using computers the people who sat at the keyboards more or less knew what they were doing. In fact they absolutely had to know what they were doing because one false move would wipe it all out. This had many more advantages than disadvantages, Quite simply, if you quit Archive with a database still open that was it. There were programs around that could recover part of the data - often in a very jumbled format - but you just learned that the software did not automatically close down the files neatly when you clicked on the quit button. this made us more careful users and taught us the value of backups. Gradually, stealthily, the computer crept out of its hiding place, in the back bedroom, behind a pile of circuit boards and old copies of Electronics Weekly or Wireless World. It paused at the door to dust itself down and adopt a new coat of shiny consumer product beige and launched a full frontal attack on your main living space. Soon it was sitting down to dinner with you all, ordering your wine and tempting you with the golden gateway to the wide world of the internet. In the course of this 25 year transition it changed its coat a few times and tried to oust all the older consumer gadgets along the way. These days we can listen to the radio on it, watch TV on it, shop on it, date on it, play music on it compose on it and store our whole collection of CDs, LPs, photos and love letters on it. What we don't do on it, or could, is to program it and that is what many of you started out wanting to do. Its new found status as all encompassing entertainments centre has also had the unfortunate effect of making it accessible to people who should not be let loose on a pocket calculator.

The reason for the above diatribe is that I recently had to set up a Windows Vista system for someone. Now, speaking as someone who used to complain to QL authors about boxes which popped up and said 'Do you really want to Quit?' you can imagine I found the whole thing annoying and deeply depressing. Here we are in the 21st century and we are treating our adult computer users to the digital equivalent of a fireguard around a fireguard around a fireguard with a big sign at the end saying 'No user serviceable parts inside'. Not only that but they won't let you into the room with the fire in it because they have locked the door and hidden the key under Bill Gates mat. But, hey, they have left a couple of windows open so the hackers can sneak in trash the furniture.

The rot, for most mainstream computer users, set in around Windows 95 and, as more and more people got computers and found more and more novel ways to wreck their systems the mighty wheels of Microsoft started trying to lock the doors to the gun cupboard to prevent the computer hillbillys shooting holes in their edifice. It is worse for them, I suppose, because they made the decision long ago that people were not able to configure their own systems so they made it so that it saves its setting when it shuts down. Good if you want the program to always open as you last left if - dreadful if you quit it because it was all in a mess. So now we have 'systems for idiots' that try to lock you out from doing anything at all. Maybe LINUX is the way ahead but they have seen the glint of a mass user base on the far horizon and have been making moves towards it. LINUX, from my, limited, experience anyway, always locked the main user out of the configuration loop but has always allowed them to log in as ROOT which allowed them to make all sorts of changes. The only trouble for me has been trying to find the time to learn enough about it to be able to make it do much. That and the fact that, as someone who is working in the mainstream computer world and who gets called on to be the 'Flying Doctor' for friends and family I have to deal with Windoze in its myriad disguises.

Roger Godley's new 'thing'

I got a surprise disk through the post a while back from Roger Godley. I have mentioned him before in this column as the man who used to do

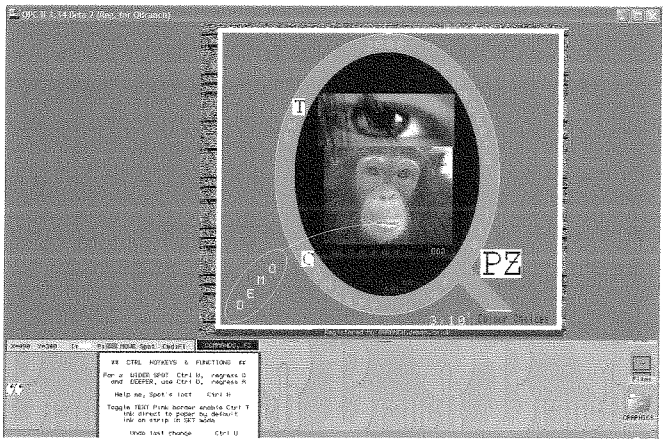
the 'cut and shut' jobs on QLs, welding two of them together and performing all manner of odd hardware experiments in a sort of Dr. Caligari way. He does also have a software side to him and he is no slouch with a bit of machine code either.

On his regular visits to the UK from his home in Spain he has often called in to see me and, over the years, has shown me some of the commercial programs he has hacked to get them to do things in either the way he wants them to be done or to run on more modern systems. His hacked versions of Quill and Archive for bigger screens are available from Quanta, I believe, and I have seen some of his revised monitor programs which work very well. Unfortunately, because these are adapted commercial programs that we no longer have access to, we are not able to distribute them.

His latest efforts dwell on the graphics and drawing area, a field where the QL is sadly lacking. Flagship programs like LINEdesign have been around for a while but are not really able to take advantage of the newer colour palettes with any ease. Page Designer 3 was abandoned by its author a while ago and had problems running on the latest systems so a new graphic drawing program would be a real boon.

Roger's efforts look pretty good, as you can see from the screen shot. It is not pointer driven, and many of the older programmers have an aversion to pointer driven programs, although, in the case of graphics, a pointer element is essential and control via the mouse is preferable. One area where the QL has always excelled is in making most of its commands available as both Key-stroke and Pointer operated. This is available to a degree on a PC but how often have you found yourself with a locked pointer and no keyboard access? The current version is locked into a 1024 x 768 definition but that can be changed.

On the whole the program looks good - albeit a long way from release in this version. I look forward to the next disk from him.



Editor's note: Roy has asked us to add an apology for the shortness of Byts of Wood in this issue due to recent family events.

Missing Bits and The Next Issue

by Geoff Wicks

Space did not allow us to include my second article on mapping and a contribution from David Denham on Ram Disks. These will appear in the next issue.

We have filled 58 pages again. Considering that this is the average size, even from the time when QL Today started (and in the early days, we had a lot more advertisers throughout the issue). We are happy to be able to produce issues containing even more information for you, compared to the past! Thank you for your help - without your support, QL Today would not exist anymore - this is addressed to both our readers and our authors!

If you want to use the Just Words! data bases to try out Hugh Room's program here are suggestions for the approximate scalings:

```
Australia and New Zealand: 50,70,-55
Austria and Switzerland: 8,4,50
Benelux 10,-5,55
British Isles: 20,-10,57
Canada: 120,-120,40
France 18,-10,45
Germany: 15,0,53
Greece: 10,10,38
Iceland: 10,-20,82
Italy: 15,0,39
Japan: 28,80,30
Scandinavia: 46,-30,60
Spain and Portugal: 15,-15,35
USA: 60,-100,10
```

QL Today Future

by Geoff Wicks

The QL Today team are currently examining the administrative, distribution and financial facilities of the magazine. This is to ensure the medium term future of QL Today in a climate where distribution costs and exchange rates are constantly changing, usually with major changes halfway through the subscription period. Many of the changes are unpredictable and we need to be in a position to react to them as quickly as possible.

To understand the situation fully it is necessary to look at the current structure of the magazine.

Although it may not be apparent to all readers the continental and UK arms of QL Today are currently two distinct entities with separate administration, distribution and financing. This arises from the very early days of the magazine where the two founding fathers were Jochen Merz and Stuart Honeyball of Miracle Systems. When Miracle Systems ceased trading QBranch took over the UK side of the operation.

Technically UK readers who order the magazine via QBranch do not have a direct subscription to QL Today. Their contract is with QBranch who then buy in the copies from the publisher.

One consequence of this is that QL Today has never had a central data base of readers and, in particular, did not have details of UK subscribers. We can see benefits in a central data base, and have now set up a simple one comprising of the names and country of our subscribers.

We hope this data base will help us to react efficiently to an increasingly complex and changing world of distribution costs.

In recent years there has been (semi-)privatisation of postal services in several countries and increasing competition to these services by the growth in the number of couriers. Both postal services and couriers are continually changing their prices, and this is not just simply by increasing them. There are also frequent changes in the categories of the various price and weight bands, which can create anomalous situations. There was a period last year, for example, when it was cheaper to post copies of the magazine to Dutch readers from Austria than it was from the Netherlands itself.

The problem is most acute with the distribution of the UK copies of the magazine. Currently these are shipped in bulk to England and posted to readers from there. We needed to examine whether it would be cheaper to post them direct from the continent. Our data base indicates that present arrangements are still the cheapest, but we cannot guarantee that this will remain so in the future.

A further complication with the UK copies is that the UK is outside the Euro zone and is thus subject to currency fluctuations. In the last year the pound has seriously weakened against the Euro - a fall of 11.4% - and most experts predict that the pound will remain weak during 2008.

The good news is that, in spite of the rise in postal rates - in the Netherlands by as much as 14% - we are able to hold the present price in Euros. However the fall in the value of the pound against the Euro means it will cost QBranch more to buy in the UK copies from the publisher, and unfortunately this means a price rise for UK readers. We can only repeat that it is not the price of QL Today that has risen, but the value of the Euro against the pound. This makes the magazine appear more expensive.

In spite of the rise in price for UK readers, we still feel QL Today gives good value for money. Editorially the magazine remains highly viable and the move to quarterly publication has made it easier for us to stick to our publishing schedules. We are blessed with an enthusiastic and skilled group of writers, and I am amazed that after nearly a quarter of a century QL-ers are still able to produce such a variety of original and interesting material.

We hope you will remain with us during volume 13, and don't worry about that number. Together we can make it a lucky QL Today year.

The QL Show Agenda

QL Meetings in Eindhoven (NL)

Saturday, 22nd of March 2008, 11:00 to 16:00
Pleincollege St. Joris, Roostenlaan 296

Thanks to the organiser, Sjef van de Molengraaf, the meetings at Eindhoven also continue in 2008. Same venue as always. J-M-S plans to be there, as always.
The dates for the other shows are already fixed, so that you can add it to your agenda.

Saturday, 14th of June 2008, 11:00 to 16:00
Pleincollege St. Joris, Roostenlaan 296

Saturday, 18th of October 2008, 11:00 to 16:00
Pleincollege St. Joris, Roostenlaan 296

Please note that the 2008 shows now start at 11:00 instead of 10!

QL Meeting in Manchester (UK)

Sat. & Sunday, 12th and 13th of April 2008
3rd Davyhulme Scout Headquarters, "The Endeavour",
Conway Road, Davyhulme, Manchester, MT41 0TE,
This is a Quanta sponsored event, but non-Quanta members are welcome as well!
Details on page 19 inside this issue!

The Next Issue

We plan to have the next issue ready for you towards the middle or end of June - maybe for the Eindhoven show. As always, it depends on how quickly we get reviews, articles etc.
We need more material, as always. The more we get and the sooner we get it, the quicker the next issue will be in your hands, and the better it will be. Hope to meet you at one of the forthcoming QL shows - your QL Today Team!